

THE POWER OF UNCERTAINTY:
ALGORITHMIC MECHANISM DESIGN IN SETTINGS OF
INCOMPLETE INFORMATION

by

Brendan Lucier

A thesis submitted in conformity with the requirements
for the degree of Doctor of Philosophy
Graduate Department of Computer Science
University of Toronto

Copyright © 2011 by Brendan Lucier

Abstract

The Power of Uncertainty:
Algorithmic Mechanism Design in Settings of
Incomplete Information

Brendan Lucier

Doctor of Philosophy

Graduate Department of Computer Science

University of Toronto

2011

The field of algorithmic mechanism design is concerned with the design of computationally efficient algorithms for use when inputs are provided by rational agents, who may misreport their private values in order to strategically manipulate the algorithm for their own benefit. We revisit classic problems in this field by considering settings of incomplete information, where the players' private values are drawn from publicly-known distributions. Such Bayesian models of partial information are common in economics, but have been largely unexplored by the computer science community.

In the first part of this thesis we show that, for a very broad class of single-parameter problems, any computationally efficient algorithm can be converted without loss into a mechanism that is truthful in the Bayesian sense of partial information. That is, we exhibit a transformation that generates mechanisms for which it is in each agent's best (expected) interest to refrain from strategic manipulation. The problem of constructing mechanisms for use by rational agents therefore reduces to the design of approximation algorithms without consideration of game-theoretic issues. We furthermore prove that no such general transformation is possible if we require mechanisms that are truthful in the stronger non-Bayesian sense of dominant strategies.

In the second part of the thesis we study simple greedy methods for resolving complex

auctions. We show that while such greedy algorithms are not truthful, they suffer very little loss in worst-case performance bounds when agents apply strategies at equilibrium, even in settings of partial information. Our analysis applies to various different equilibrium concepts, including Bayes-Nash equilibrium, regret-minimizing strategies, and asynchronous best-response dynamics. Thus, even though greedy auctions are not truthful, they may be appropriate for use as mechanisms under the goal of achieving high social efficiency at equilibrium. Moreover, we prove that no algorithm in a broad class of greedy-like methods can be used to create a deterministic truthful mechanism while retaining a non-trivial approximation to the optimal social welfare.

Our overall conclusion is that while full-information models of agent rationality currently dominate the algorithmic mechanism design literature, a relaxation to settings of partial information is well-motivated and provides additional power in solving central problems in the field.

Acknowledgements

I am deeply indebted to my graduate supervisors Allan Borodin and Mike Molloy. They not only provided me with a broad and deep perspective of theoretical computer science, but also transformed a simple graduate program into a fruitful and exciting journey into the world of research.

To all those who helped with the editing of this thesis, including those who listened politely while I thought out loud in response to an innocent question about how things were coming along, I thank you. I would especially like to thank Charles Rackoff for his many insightful comments and suggestions; his input substantially improved the presentation of this work.

I am extremely grateful to my fellow graduate students and collaborators for providing me with such a positive research experience. An especially large fraction of this thanks goes to my coauthors Jason Hartline and Nicole Immorlica, with whom much of the content of this thesis was developed. My visits to Northwestern were highlights of my time as a graduate student.

This work was made possible by the financial support of NSERC and the University of Toronto; I am grateful for their generosity.

Finally, my heartfelt thanks go out to my family and friends, without whose unwavering support and encouragement I could not possibly have completed this work. Thank you.

Contents

1	Introduction	1
1.1	Roadmap and Contributions	5
2	Background	11
2.1	Mechanisms	11
2.2	Social Welfare and Other Objectives	15
2.3	Truthful Mechanisms for Social Welfare	17
2.4	Equilibria and Price of Anarchy	20
2.5	Single-Parameter Mechanism Design	22
2.6	Combinatorial Allocation Problems	25
2.6.1	Critical Prices	26
2.6.2	Input Representation	27
2.6.3	Examples	28
2.7	Related Work	30
I	Single-Parameter Problems	37
3	Single-Parameter Bayesian Incentive Compatibility	38
3.1	Preliminaries	42
3.2	The Approach	43
3.2.1	Ironing via Resampling	45

3.2.2	The Ironed Algorithm	47
3.2.3	Computational Issues	49
3.2.4	Comparison with Myerson’s Mechanism	50
3.3	A Black-Box Reduction	52
3.3.1	Computing Payments	54
3.3.2	Sampling and Approximate Truthfulness	55
3.3.3	Exact Truthfulness	60
3.3.4	Multiplicative Error	66
3.4	Limits of our Approach	75
3.4.1	Worst-Case Approximations	75
3.4.2	Beyond Social Welfare	76
4	Dominant Strategy Transformations	81
4.1	Preliminaries	83
4.2	Impossibility of Lossless Transformations	84
4.2.1	Construction	85
4.2.2	Analysis	88
4.3	Randomized Transformations	94
II	Combinatorial Auction Problems	95
5	Equilibria of Greedy Auctions	96
5.1	Preliminaries	103
5.1.1	Feasible Allocation Problems	103
5.1.2	Greedy Allocation Rules	103
5.1.3	Mechanisms and Payment Methods	104
5.1.4	Smoothness	105
5.2	Strong Loser-Independence	107

5.3	First-Price Mechanisms	110
5.3.1	Warmup: Pure Nash Equilibria	112
5.3.2	Existence of Pure Nash Equilibria	113
5.3.3	Bayes-Nash Equilibria	114
5.3.4	Correlated Types	122
5.4	Critical-Price Mechanisms	124
5.4.1	Calculating Critical Prices	125
5.4.2	Bayes-Nash Equilibria	126
5.5	Combining Mechanisms	129
5.6	Tightening Results for Special Cases	132
5.7	Applications	134
6	Repeated Combinatorial Auctions	137
6.1	Preliminaries	142
6.1.1	Repeated Auctions	142
6.1.2	Regret Minimization	143
6.2	A Mechanism for Regret-Minimizing Bidders	144
6.2.1	Resilience to Irrational Strategies	149
6.2.2	Importance of Strong Loser-Independence	150
6.3	Best-Response Agents	151
6.3.1	The Approach	153
6.3.2	A Mechanism for s -CAs	157
6.3.3	A Mechanism for General CAs	162
6.4	Removing Assumptions	169
7	Truthful Priority Algorithms	170
7.1	Preliminaries	173
7.1.1	Critical Prices	173

7.1.2	Priority Algorithms	175
7.2	Truthful Priority Algorithms	176
7.2.1	Sets as Items	177
7.2.2	Elementary bids as items	184
7.2.3	Bidders as Items	185
8	Conclusions	191
	Bibliography	193

List of Figures

3.1	A non-monotone ironing	46
3.2	A monotone ironing	47
3.3	A sample virtual valuation function	51
3.4	An allocation rule for which ironing does not preserve worst-case approx- imations	76
4.1	Construction of an algorithm that resists dominant strategy transformations	86
6.1	A mechanism for regret-minimizing bidders	145
6.2	A best-response mechanism for the s -CA problem	157
6.3	A best-response mechanism for the general combinatorial allocation problem	163

Chapter 1

Introduction

Recent years have been marked by greatly increased interaction between computer systems and society at large in everyday tasks. It is now commonplace to use Internet-enabled applications for socializing, shopping, entertainment, and more. As computer systems become increasingly integrated into essential facets of everyday life, we must become increasingly aware of how implementation details can affect - and be manipulated by - the end users. The result has been a surge of interest in the social and economic aspects of computing tasks. Algorithms for social networks, implementations of online auctions, Internet protocols for selfish routers; these are some of the many examples of algorithmic topics in which the nuances of real-world user behaviour have a profound impact on system performance. The added difficulty in such systems is that they depend upon interaction with participants whose behaviour is motivated by their own goals, rather than those of a designer. Relevant solutions must therefore merge the computational considerations of computer science with the game-theoretic insights of economics.

Understanding the interaction between computational limitations and individual incentives is crucial to developing a theory of large-scale systems that interact directly with human users. The area of research that addresses this intersection is known as *algorithmic mechanism design*. This field concerns the design of algorithms in settings

where the input is controlled by selfish agents. These agents may strategically misreport values to manipulate an algorithm's outcome. The standard example is an auction: the participants' values for the good(s) being sold is private information unknown to the auctioneer. Thus, when an auctioneer implements an auction, she must be sensitive to the fact that participants will behave in the interest of obtaining valuable goods at low costs, rather than behaving as the auction designer wishes.

As an algorithmic mechanism designer, one generally wishes to design an algorithm that optimizes some goal, such as maximizing revenue or social efficiency. This optimization must be performed in spite of two obstacles: first, the system must be computationally efficient; and second, it must resist rational manipulation by the participants. While economic theory provides many tools for dealing with the latter problem (see, for example, [55], [73], and chapter 9 of [77]), many of the known techniques are not computationally efficient and therefore infeasible in very large problem instances (such as large-scale auctions executed over the internet). The primary goal of algorithmic mechanism design, then, is to determine to what extent the insights of economic theory can be preserved when we limit ourselves to mechanisms that can be resolved by computationally efficient algorithms.

An important aspect of a mechanism design problem is the manner in which agent behaviour is modelled. We assume that individual participants behave rationally, but how precisely do we model "rational" behaviour? This, in turn, determines our solution concept: the properties we require of an algorithm in order for us to conclude that it generates a desirable outcome when used by strategic agents. Much of the existing literature in algorithmic mechanism design has focused on one particular solution concept: truthfulness in dominant strategies (or simply *truthfulness*). A mechanism is *truthful* if it is always in each agent's best interest to participate honestly, without manipulation, regardless of how others behave. This is a very strong solution concept, motivated by its synergy with worst-case approximation analysis and the classic result in economic mech-

anism design that it is always possible to design a truthful mechanism that maximizes social efficiency if we ignore computational issues [86, 24, 44]. Much of the recent work in algorithmic mechanism design has been directed at exploring the effects of computational issues on this result, culminating in impossibility results that show that many natural goals (such as matching the social welfare guarantees that can be obtained by algorithms without game-theoretic concerns) cannot be achieved by truthful algorithms [80].

While such negative results are discouraging, they need not signify the end of this line of research. Dominant strategy truthfulness is not the only solution concept used in the theory of economic mechanism design. Indeed, the concept of truthfulness carries with it an extreme degree of certainty on the part of the auction participants: each player must be completely confident that it is optimal to not strategically manipulate the mechanism, even in hindsight after the strategic choices of all other players have been revealed. That is, even in a full-information setting where each player knows everything about his opponents' choices in advance, it is required that this certainty cannot be leveraged into a strategic advantage.

An alternative approach, common in economic mechanism design, is to model the information that agents have about each others' behaviour, then assume that agents behave rationally subject to this (partial) information. That is, one might wish to assume that agents are neither completely informed nor completely ignorant of how the other participants behave, but rather have some partial information about their opponents. This is typically modelled in a Bayesian manner, introduced by Harsanyi [46]: roughly speaking, one assumes that there is some publicly-known distribution over agent types (i.e. private values). This partial information represents the collective experience and/or market data available to the bidders, which is well-motivated in large-scale high-stakes auctions. Such partial information models are ubiquitous in economic theory (see, for example, surveys by Jackson [55] and Myerson [73]), but have been largely unexplored by the computer science community.

As a further alternative, one need not limit oneself to mechanisms in which truthfulness is a dominant strategy. It may be perfectly acceptable for a system to admit strategic manipulation, so long as the designer’s objectives are met after such manipulation occurs. To this end, we take the standard game-theoretic view that rational agents will apply bidding strategies that are at *equilibrium*, meaning (roughly) that no single agent can improve his outcome by modifying his strategy. Thus, while agents may not know for certain that their behaviour will be optimal in retrospect, they will nevertheless choose the best strategy given the partial information that they do have available. Our goal as a mechanism designer, then, would be to design a mechanism that achieves good performance at *every* equilibrium of bidding behaviour¹. The promise of such a solution concept is that simple algorithms, disregarded previously due to their lack of truthfulness, may be redeemed as possible candidates for mechanisms if they perform well at equilibrium.

In this thesis, I explore the design of computationally efficient mechanisms using these alternative partial-information solution concepts. Broadly speaking, my motivating questions are as follows:

1. Under what conditions can an approximation algorithm be made truthful in partial information settings?
2. When can an algorithm be converted into a mechanism that achieves high social welfare at every equilibrium?
3. Does the adoption of partial information and/or equilibrium solution concepts afford additional power over dominant strategy truthfulness?

I will demonstrate that certain open problems that are central to algorithmic mech-

¹We note that this solution concept is different from the notion of *implementation at equilibrium*, in which one would like a specific outcome to occur at an equilibrium of the mechanism [55]. In our case we care only about the approximation factor of an outcome, but we require that this approximation factor hold at every equilibrium.

anism design can be resolved in partial information settings or when using equilibrium solution concepts. Moreover, the mechanisms I will present have desirable properties (such as generality of construction or natural greedy-like allocation rules) that provably cannot be obtained if one insists upon dominant strategy truthfulness. Our conclusion is that exploring solution concepts beyond dominant strategy truthfulness is well-motivated and enables us to overcome some of the apparent obstacles to computationally efficient mechanism design.

1.1 Roadmap and Contributions

In the first part of my thesis, I will consider the design of mechanisms for single-parameter social welfare problems. In this class of problems, each agent has a private value for receiving some sort of service; the designer's goal is to choose how to distribute service in order to maximize the total value achieved, subject to arbitrary feasibility and/or cost constraints. This class of problems includes combinatorial auctions, facility location problems, public good problems, and many others.

A large body of work in algorithmic mechanism design has dealt with designing truthful mechanisms for problems in this class. The current state of the art is that there are no general techniques for converting (non-truthful) approximation algorithms into truthful algorithms without loss. Generally, one considers a specific problem and attempts to design a truthful algorithm that matches the approximation ratio of the best-known non-truthful algorithm.

Our results differ from this line of work in that we instead consider a setting of partial information. In a partial information setting, each agent's (private) value for service is drawn from a publicly-known distribution. These distributions represent the incomplete information available to the participants. We then say that a mechanism is *Bayesian incentive compatible* if each agent maximizes his *expected* benefit by reporting his value

truthfully, under the assumption that other agents also report their values truthfully. This is the natural analog of truthfulness in a Bayesian setting. Our goal will be to design mechanisms that are truthful in this Bayesian sense.

In Chapter 3 we provide our main positive result. We demonstrate that *any* algorithm for *any* single-parameter social welfare problem can be converted into a Bayesian incentive compatible mechanism without loss of expected performance. That is, if one wishes to design a mechanism for a single-parameter problem, it is sufficient to design an algorithm without consideration of agent incentives; this algorithm can then be transformed into a mechanism that is truthful in the Bayesian sense. The original algorithm need not be a worst-case approximation algorithm, but can be an arbitrary heuristic tailored to the expected distribution over inputs. Moreover, our result is constructive: we provide a polynomial-time transformation that, given black-box access to an algorithm \mathcal{A} and the distributions from which the agents' values are drawn, returns a Bayesian incentive compatible mechanism with expected performance matching that of algorithm \mathcal{A} . This black-box transformation proceeds by sampling, which introduces an additive loss in social welfare that can be made arbitrarily small. Moreover, our transformation applies to a very broad class of problems that include arbitrary costs for outcomes, and requires no information about the specific constraints of the problem to be solved. Thus, in a Bayesian setting, the problem of designing a truthful mechanism reduces to the problem of designing an approximation algorithm without consideration of incentive issues. This chapter is based upon joint work with Jason Hartline [48].

In Chapter 4, we ask whether a transformation like the one described in Chapter 3 is possible if we insist upon dominant strategy truthfulness in non-Bayesian settings. That is, can we convert arbitrary approximation algorithms into dominant strategy truthful mechanisms in a black-box manner? Formally, we'll say a polytime transformation \mathcal{T} is an algorithm that takes as input an algorithm \mathcal{A} and input declaration from the bidders and returns an outcome. That is, we can think of $\mathcal{T}(\mathcal{A})$ as an algorithm for

social welfare maximization; we think of this as the algorithm \mathcal{A} transformed by \mathcal{T} . We say that \mathcal{T} is a (dominant strategy) truthful transformation if $\mathcal{T}(\mathcal{A})$ is (dominant strategy) truthful for all \mathcal{A} . Our question then becomes: does there exist a truthful transformation that (approximately) preserves worst-case approximation factors? We show that the answer is no: it is not possible to have a single black-box transformation that converts algorithms into dominant strategy truthful mechanisms without a very large loss in worst-case approximation factor for some algorithms. This result holds for deterministic mechanisms and for randomized mechanisms that are universally truthful. The relaxation from dominant strategy truthfulness to Bayesian incentive compatibility therefore provably gives us additional power in our ability to reduce mechanism design to algorithm design. This chapter is based upon joint work with Nicole Immorlica [54].

In the second part of my thesis, I will focus on the class of combinatorial auction problems, in which there is a set M of m objects for sale to be distributed among the auction participants, subject to feasibility constraints. Unlike the problems considered in the first part of the thesis, combinatorial allocation problems need not be single-parameter. This class of problems includes combinatorial auctions, multi-unit combinatorial auctions, unsplitable flow routing problems, profit-maximizing job scheduling, and others.

I wish to consider a class of *greedy* algorithms for these allocation problems as candidates for mechanism allocation rules. In general, greedy algorithms are not truthful for combinatorial allocation problems [64]. However, greedy mechanisms do have a number of advantages that are difficult to overlook. First, despite their simplicity, greedy algorithms are known to obtain asymptotically tight (or nearly tight) approximation bounds for many CA problems, subject to standard complexity constraints [64, 71, 18, 7, 21]. In addition to their theoretical performance bounds, greedy algorithms have been found to perform surprisingly well in experiments with realistically distributed data [43]. Moreover, the simplicity of greedy methods yields numerous benefits for practical auction design: many of the standard greedy algorithms are deterministic and easy for partici-

pants to understand, so that a bidder can quickly determine how much they would need to bid in order to obtain a particular outcome. Such transparency is important, as agents are unlikely to participate in an auction when they do not understand the rules for determining winning bids [5]. Indeed, many auctions used in practice apply greedy methods, despite the fact that they may not be incentive compatible (e.g. the generalized second price auction for sponsored search advertisements [38]). Our observation, then, is that simple mechanisms (and greedy methods in particular) seem to be good candidates for auctions due to considerations beyond truthfulness, such as observed performance, ease of public understanding, and perceived fairness.

It is for the above reasons that we wish to analyze the game-theoretic properties of greedy mechanisms, beyond the question of whether or not they are dominant strategy truthful. In Chapter 5, we consider the performance of greedy mechanisms - using standard payment schemes - when agents play strategies at equilibrium. What we find is that, in both full-information and partial-information settings, a greedy approximation algorithm generates a mechanism whose worst-case performance at *every* equilibrium nearly matches that of the original algorithm. That is, a c -approximation algorithm yields a mechanism with *price of anarchy* at most $c + o(c)$. We consider both first-price mechanisms (where each agent pays their bid) and critical-price mechanisms (where each agent pays the smallest bid that would win their allocation), and give tight bounds on the equilibrium performance of each. The results of this chapter are based upon joint work with Allan Borodin [67].

Just as in Chapter 3, we are considering a setting of incomplete information to model agent behaviour. However, unlike the case of incentive compatibility, the introduction of Bayesian types serves to increase the difficulty of our mechanism design problem. Intuitively speaking, a bound on the performance of equilibria in the full-information setting necessarily depends on the (strong) assumption that agents have perfect knowledge of each others' types. To escape this strong assumption, we instead bound performance

at equilibria that arise when agents have only partial type information. Moreover, our mechanism will not make use of the distributional information, but is rather a single deterministic mechanism that achieves the given bounds for every distribution of agent types.

We go on to show that our construction is robust in that it holds for a number of perturbations of our equilibrium concept. For instance, we show that if, in addition to the rational participants, we *add* an arbitrary number of irrational bidders who cannot be assumed to bid at equilibrium, this does not decrease the social welfare obtained by the mechanism (relative to what would have been obtained had the irrational bidders not participated). Additionally, the performance of our mechanism degrades gracefully if we assume that rational agents play at *approximate* equilibrium. In particular, if each agent obtains utility that is within a factor of $\gamma \geq 1$ from the optimum (given the strategies applied by the other agents) then our bound on the performance of a critical-price mechanism based upon a c -approximate greedy algorithm becomes $c + \gamma$.

The results of Chapter 5 assume that agents will always find and play strategies at equilibrium. For certain complex problems this may be an overly strong assumption, especially since finding equilibria is known to be computationally difficult in general [26]. In Chapter 6 we relax our equilibrium concepts and analyze the performance of greedy mechanisms under alternative models of agent behaviour. We consider a repeated variant of the combinatorial allocation problem, which models processes by which agents can iteratively update their strategies in response to their opponents. We demonstrate that if we modify our mechanisms slightly, agents can easily find regret-minimizing bidding strategies using standard techniques from learning theory [57]. We then show that if agents minimize their regret, the average social welfare obtained after polynomially many rounds again yields a $c + o(c)$ approximation to the optimal social welfare. We also show that in the special case of the general combinatorial auction problem, a modification of our mechanism obtains an $O(c)$ approximation to the optimal social welfare if agents do

not minimize regret, but rather play iterated best-response strategies. These results taken together show that it is possible to guarantee high social welfare on average over many repetitions of an auction, even when agents are computationally bounded and cannot be assumed to converge to equilibrium. The results of this chapter appeared as [66].

In Chapter 7, we ask whether there is any *truthful* greedy mechanism for the combinatorial auction problem that obtains performance close to the performance of the above mechanisms at equilibrium. We will show that the answer is no: there is no truthful mechanism for the combinatorial auction problem with a greedy allocation rule that yields social welfare approaching that of the standard greedy algorithms. The notion of “greedy” auction we use is quite broad: we consider the class of all greedy *priority algorithms* for an auction problem [17]. Roughly speaking, a priority algorithm for a combinatorial allocation problem treats an input instance as a collection of input items, which are ranked according to some (possibly adaptive) ranking function. The algorithm then repeatedly considers the highest-ranked feasible bid and accepts it. We prove that, under this model, any truthful greedy priority algorithm for a combinatorial auction problem must have an extremely poor approximation ratio (i.e. linear in the number of objects for sale). The gap implied by this negative result is extreme: the theorem applies to cases in which a simple greedy algorithm obtains a 3-approximation, whereas a linear approximation can be achieved trivially (e.g. by allocating all objects to a single bidder). In short, there is a huge gap in the power of greedy algorithms and greedy truthful algorithms for combinatorial auctions. The results of this chapter are based upon joint work with Allan Borodin [16].

Our conclusion is that relaxing the solution concept from dominant strategy truthfulness to performance at equilibrium enables us to construct conceptually simple greedy mechanisms that generate high social welfare. Thus, if one is looking to design an auction with the benefits of a simple and transparent resolution rule, it can be advantageous to relax the desideratum of truthfulness to an equilibrium-based solution concept.

Chapter 2

Background

2.1 Mechanisms

We begin with a brief overview of relevant concepts in mechanism design. We will limit our attention to a particular subclass of mechanisms¹ upon which this thesis is focused; for a more thorough treatment of mechanism design from a computer science perspective see Chapter 9 of [77]. We will use the notational convention that boldface variables denote vectors and that a subscript of the form “ $-i$ ” denotes all entries of a vector excluding entry i , so that (for example) $\mathbf{v} = (v_i, \mathbf{v}_{-i})$.

A *mechanism* describes an interaction protocol between a central authority and n agents. The goal of this interaction is to choose an *allocation* from a set O of possible outcomes. For notational convenience, we will suppose that $O = O_1 \times \dots \times O_n$, so that we can think of the mechanism as choosing an allocation $x_i \in O_i$ for each agent i . The mechanism also determines *payments* $p_1, \dots, p_n \in \mathbb{R}$, where we think of p_i as an amount that agent i should pay to the mechanism. We note that payments will be non-negative in all of the mechanisms considered in this thesis.

Each agent i is assumed to have a *valuation function* $v_i : O_i \rightarrow \mathbb{R}$ that assigns a value

¹Specifically, we will consider only direct-revelation mechanisms with payments, and we will assume that agents have quasi-linear utilities.

to each allocation. We refer to this function as the *type* of agent i . Write V_i for the space of allowable types for agent i , and $V = V_1 \times \dots \times V_n$ for the space of possible type profiles. We generally think of an agent's type as being private information known only to himself. In this thesis we will restrict attention to cases in which valuation functions take on only non-negative values. If a mechanism generates allocation $\mathbf{x} \in O$ and payment vector $\mathbf{p} \in \mathbb{R}^n$, then the *utility* of agent i is $u_i = v_i(x_i) - p_i$. We think of utility as the overall benefit that an agent derives from participating in the mechanism. We view each agent as a rational entity who wishes to maximize his or her own utility.

A mechanism proceeds by asking each agent to declare his or her type as a sealed bid. The mechanism then reveals these type declarations simultaneously and decides upon an allocation profile $\mathbf{x} \in O$ and a payment profile $\mathbf{p} \in \mathbb{R}^n$. We will therefore describe the behaviour of a mechanism by an *allocation rule* $\mathbf{x}: V \rightarrow O$ that generates an allocation and a *payment rule* $\mathbf{p}: V \rightarrow \mathbb{R}^n$ that determines payments. That is, given allocation rule $\mathbf{x}(\cdot)$ and payment rule $\mathbf{p}(\cdot)$, a mechanism is described by the pair $\mathcal{M} = (\mathbf{x}(\cdot), \mathbf{p}(\cdot))$. We note that a mechanism may be randomized, in which case we view $\mathbf{x}(\mathbf{v})$ and $\mathbf{p}(\mathbf{v})$ as random variables.

Crucially, it is not required that agents report their types truthfully to the mechanism. We will tend to write \mathbf{v} for the true types of the agents and \mathbf{d} for a declared type profile (which may not be equal to \mathbf{v}) when this distinction is important. Given a fixed type profile \mathbf{v} and a fixed mechanism \mathcal{M} with allocation rule $\mathbf{x}(\cdot)$ and payment rule $\mathbf{p}(\cdot)$, we will write² $u_i(\mathbf{d})$ for the utility obtained by agent i when the agents declare type profile \mathbf{d} to mechanism \mathcal{M} . that is, $u_i(\mathbf{d}) = v_i(x_i(\mathbf{d})) - p_i(\mathbf{d})$.

To this point we have described mechanisms purely in terms of the outcomes they generate on different inputs; that is, as functions $\mathbf{x}(\cdot)$ and $\mathbf{p}(\cdot)$. In algorithmic mechanism

²In this and much of our subsequent notation, we do not explicitly denote the dependency on the mechanism \mathcal{M} and/or true type profile \mathbf{v} ; these are generally taken to be implicit when clear from context. Additional notation will be introduced in the (rare) cases where there are multiple mechanisms or type profiles under consideration.

design we are also concerned with the computational properties of an implementation of these functions. To this end, we define an *allocation algorithm* as an algorithm \mathcal{A} that takes as input a valuation profile $\mathbf{v} \in V$ and returns an allocation profile $\mathbf{x} \in O$. Similarly, a *payment algorithm* \mathcal{P} takes as input a valuation profile $\mathbf{v} \in V$ and returns a payment profile $\mathbf{p} \in \mathbb{R}^n$. An allocation algorithm \mathcal{A} and payment algorithm \mathcal{P} together describe the implementation of a mechanism $\mathcal{M} = (\mathcal{A}, \mathcal{P})$ in which \mathcal{A} is used to compute allocations and \mathcal{P} is used to compute payments. We will tend to switch between the algorithmic viewpoint and the functional viewpoint as convenient, so that for a fixed mechanism \mathcal{M} we will write \mathcal{A} for the algorithm that determines its allocations and $\mathbf{x}(\cdot)$ for the allocation rule implemented by \mathcal{A} .

We now proceed to define certain useful properties of mechanisms. A mechanism is said to be (ex post) *individually rational* if any agent that reports her type truthfully always obtains non-negative utility, regardless of the declarations of the other agents. That is, for all i , all v_i , and all \mathbf{d}_{-i} , $u_i(v_i, \mathbf{d}_{-i}) \geq 0$. A mechanism is said to have *no positive transfers* if it never charges negative payments, which is interpreted as giving a payment to an agent. That is, a mechanism with no positive transfers satisfies $p_i(\mathbf{d}) \geq 0$ for all \mathbf{d} . All of the mechanisms we consider in this thesis will have non-negative payments.

A (deterministic) *bidding strategy* for agent i is a function $b_i: V_i \rightarrow V_i$, where we think of $b_i(v_i)$ as the declaration suggested by strategy b_i when the agent's true type is v_i . For a deterministic mechanism \mathcal{M} , strategy b_i *weakly dominates* strategy b_i' for agent i and mechanism \mathcal{M} if b_i always generates at least the utility of b_i' , given any declaration of the other agents. That is, for all v_i and all \mathbf{d}_{-i} ,

$$u_i(b_i(v_i), \mathbf{d}_{-i}) \geq u_i(b_i'(v_i), \mathbf{d}_{-i}).$$

More generally, strategy b_i weakly dominates strategy b_i' in a (possibly randomized) mechanism \mathcal{M} if it weakly dominates b_i' in every deterministic mechanism in the support of \mathcal{M} . Strategy b_i *strictly dominates* b_i' if, in addition to weak domination, there exists

a declaration of the other agents such that the utility of b_i is strictly greater than the utility of b_i' . A strategy b_i is *undominated* if there is no b_i' that strictly dominates b_i . A strategy b_i is a *dominant strategy* if it weakly dominates all other strategies. Note that every dominant strategy is undominated, but not vice-versa.

Definition 2.1.1 (Dominant strategy truthfulness) *A mechanism \mathcal{M} is dominant strategy truthful or dominant strategy incentive compatible if the truth-telling strategy $b_i(v_i) = v_i$ is a dominant strategy for each agent i . In other words, each agent always maximizes his utility by declaring his true type, regardless of the declaration of the other agents. That is, for all i , all true types v_i for agent i , and all possible declaration profiles \mathbf{d} ,*

$$u_i(v_i, \mathbf{d}_{-i}) \geq u_i(d_i, \mathbf{d}_{-i}).$$

We will say that an allocation rule $\mathbf{x}(\cdot)$ is dominant strategy truthful if there exists a payment rule $\mathbf{p}(\cdot)$ such that the mechanism $\mathcal{M} = (\mathbf{x}, \mathbf{p})$ is dominant strategy truthful. Similarly, an allocation algorithm is dominant strategy truthful if it implements an allocation rule that is dominant strategy truthful.

There are two competing notions of truthfulness for randomized mechanisms. A randomized mechanism \mathcal{M} is called *universally truthful* if every deterministic mechanism in the support of \mathcal{M} is dominant strategy truthful. Note that this notion of truthfulness corresponds to dominant strategy incentive compatibility as defined above. That is, it is a dominant strategy for an agent to report his type truthfully in a universally truthful mechanism. By contrast, a mechanism is *truthful in expectation* if each agent always maximizes his *expected* utility by declaring his true type, for any declaration of the other agents, with expectation taken over any randomness in the mechanism. That is, for all i , all v_i , and all \mathbf{d} ,

$$\mathbf{E}[u_i(v_i, \mathbf{d}_{-i})] \geq \mathbf{E}[u_i(d_i, \mathbf{d}_{-i})].$$

We will sometimes refer to a mechanism that is truthful in expectation as being *ex post*

incentive compatible. Note that ex post incentive compatibility and dominant strategy incentive compatibility are equivalent for deterministic mechanisms.

We will say that an allocation rule $\mathbf{x}(\cdot)$ or an allocation algorithm \mathcal{A} is universally truthful (resp. truthful in expectation) if it can be paired with a payment rule $\mathbf{p}(\cdot)$ such that the resulting mechanism is universally truthful (resp. truthful in expectation).

In a setting of partial information, we assume that agent types are drawn at random from a distribution \mathbf{F} . It is standard to assume that types are distributed independently, so that $\mathbf{F} = F_1 \times \dots \times F_n$. The following is the standard notion of truthfulness in partial information settings.

Definition 2.1.2 (Bayesian incentive compatibility) *We say that a mechanism \mathcal{M} is Bayesian incentive compatible (BIC) with respect to type distribution \mathbf{F} if truth-telling maximizes the expected utility of each agent, under the assumption that other agents report truthfully and their types are distributed according to \mathbf{F} . The expectation is taken with respect to the distribution over agent types and also over any randomization in the mechanism. That is, for all i , any true type v_i , and any type declaration d_i ,*

$$\mathbf{E}_{\mathbf{v}_{-i}}[u_i(v_i, \mathbf{v}_{-i})] \geq \mathbf{E}_{\mathbf{v}_{-i}}[u_i(d_i, \mathbf{v}_{-i})].$$

As with the other notions of truthfulness, we will say that an allocation rule $\mathbf{x}(\cdot)$ or an allocation algorithm \mathcal{A} is Bayesian incentive compatible if there exists a payment rule $\mathbf{p}(\cdot)$ such that the resulting pair is a mechanism that is Bayesian incentive compatible.

2.2 Social Welfare and Other Objectives

The *Social welfare* or *social efficiency* of an outcome represents the benefit of that outcome for the group of participants. Suppose that the mechanism incurs a cost $c(\mathbf{x})$ for generating outcome \mathbf{x} . Then given types $\mathbf{v} \in V$, the social welfare of allocation $\mathbf{x} \in O$ is $SW(\mathbf{x}, \mathbf{v}) = \sum_i v_i(x_i) - c(\mathbf{x})$. Note that this is the sum of utilities for all participants

including the mechanism itself, since payments are simply transferred from the agents to the mechanism. We will allow allocation \mathbf{x} to be a random variable, in which case $SW(\mathbf{x}, \mathbf{v})$ denotes the expected social welfare of \mathbf{x} . We note that for many problems of interest the cost function $c(\cdot)$ will be identically zero. Given a type profile \mathbf{v} , the optimal social welfare is written $SW_{OPT}(\mathbf{v}) = \max_{\mathbf{x} \in \mathcal{O}} SW(\mathbf{x}, \mathbf{v})$.

In this thesis we will be focused on the problem of designing mechanisms that maximize (or approximately maximize) social welfare. For such problems, we view the mechanism as a central authority that is attempting to align the incentives of a group of individuals, who would be unable to agree upon a globally optimal outcome due to conflicts in their individual desires. As we are interested in problems in algorithmic mechanism design, it will be important for us to consider computational constraints in our method for maximizing welfare. Recall that we think of an *allocation algorithm* \mathcal{A} as an implementation of an allocation rule. In general, when \mathcal{A} is some implicitly understood algorithm, we will tend to write $\mathbf{x}(\cdot)$ for the allocation rule of \mathcal{A} . We will also sometimes write $\mathcal{A}(\mathbf{v})$ to mean $\mathbf{x}(\mathbf{v})$ when notationally convenient. We say that \mathcal{A} is a worst-case c -approximation algorithm if $SW(\mathcal{A}(\mathbf{v}), \mathbf{v}) \geq \frac{1}{c} SW_{OPT}(\mathbf{v})$ for all types \mathbf{v} . Note that we take the convention that $c \geq 1$, but one could equivalently define this as a $\frac{1}{c}$ -approximation. Also, our definition of a worst-case approximation algorithm applies even if \mathcal{A} is randomized, in which case we recall that $SW(\mathcal{A}(\mathbf{v}), \mathbf{v})$ denotes the expected social welfare of \mathcal{A} on input \mathbf{v} .

In a Bayesian setting where agent types are drawn from a distribution \mathbf{F} over V , we will be concerned with the expected social welfare of an algorithm \mathcal{A} , given by $\mathbf{E}_{\mathbf{v} \sim \mathbf{F}}[SW(\mathcal{A}(\mathbf{v}), \mathbf{v})]$. We will sometimes write \mathcal{A} for the expected welfare of algorithm \mathcal{A} under (implicit) distribution \mathbf{F} , where the meaning is clear from context. We say that \mathcal{A} is a *Bayesian c -approximation* if $\mathcal{A} \geq \frac{1}{c} \mathbf{E}_{\mathbf{v}}[SW_{OPT}(\mathbf{v})]$.

Social welfare maximization is not the only goal studied in algorithmic mechanism design. Another common goal is to design mechanisms for job scheduling on parallel

machines, with the goal of minimizing the makespan (i.e. latest completion time). For such a problem we think of the jobs as objects to be assigned and the machines as agents who can process jobs, where the speed at which each machine can process jobs is private information. Two commonly-studied models are related machine scheduling and unrelated machine scheduling. In the unrelated machines model, each machine has an arbitrary (private) processing time for each possible job. In the related machines model, each machine has only a single private parameter representing its speed, and processing time for a job is the job length divided by machine speed. In both models, the mechanism elicits reports of the private information and attempts to schedule jobs with the goal of minimizing the latest finishing time.

Another extremely important objective is that of revenue maximization. The revenue of a mechanism is given by $\sum_i p_i(\mathbf{d})$. Revenue maximization is fundamentally different from welfare maximization and makespan minimization, as it is not a (direct) function of the allocation generated by the agents but rather the payments extracted from the agents. Thus, while we can think of a social welfare maximizing or makespan-minimizing mechanism as helping a group of agents to choose a globally beneficial outcome, we think of a revenue-maximizing mechanism as representing the interests of an outside player (e.g. the auctioneer) whose incentives are directly opposed to those of the agents.

In this thesis we will be concerned exclusively with mechanisms for social welfare maximization, though we do explore some of the limitations of our techniques for makespan minimization problems in Chapter 3.

2.3 Truthful Mechanisms for Social Welfare

In this section we review some classic results from mechanism design regarding the construction of mechanisms without consideration of computational issues.

The VCG Mechanism

We first describe the well-known VCG mechanism and show that it truthfully implements any allocation rule that is optimal with respect to social welfare. Suppose that, for every declaration \mathbf{d} , allocation rule \mathcal{A} returns an allocation that optimizes social welfare. The VCG payment scheme for \mathcal{A} is as follows. Fix \mathbf{d} , and suppose X_1, \dots, X_n is an optimal allocation for \mathbf{d} . Choose $i \in [n]$, and suppose Y_1, \dots, Y_n is an allocation that maximizes social welfare *excluding agent i* ; that is, an allocation that maximizes $\sum_{j \neq i} d_j(Y_j)$. Then on input \mathbf{d} , the VCG payment for agent i is:

$$p_i = \sum_{j \neq i} d_j(Y_j) - \sum_{j \neq i} d_j(X_j).$$

The VCG mechanism combines the optimal allocation rule \mathcal{A} with the VCG payment scheme. One can interpret the VCG payment scheme as “internalizing the externalities” of the players: each agent pays the amount that the social welfare of the other agents decreased due to his presence in the auction. The statement and proof of the Vickrey-Clarke-Groves Theorem below is classical; see [69].

Theorem 2.3.1 (Vickrey-Clarke-Groves) *The VCG mechanism is dominant strategy truthful, individually rational, and has non-negative payments.*

Proof: Suppose the true types of the agents are \mathbf{v} . Fix $i \in [n]$ and \mathbf{d}_{-i} . As above, let Y_1, \dots, Y_n denote an allocation maximizing $\sum_{j \neq i} d_j(Y_j)$. If agent i declares d_i , then his utility is $v_i(X_i) + \sum_{j \neq i} d_j(X_j) - \sum_{j \neq i} d_j(Y_j)$, where we recall that X_1, \dots, X_n is the optimal allocation for declaration \mathbf{d} . Since the last sum is not affected by agent i 's declaration, agent i will maximize his utility by choosing d_i to maximize $v_i(X_i) + \sum_{j \neq i} d_j(X_j)$ (recall \mathbf{X} depends on d_i). By our choice of X_1, \dots, X_n , this expression is maximized if agent i declares $d_i = v_i$, so the mechanism is truthful.

By our choice of Y_1, \dots, Y_n in the definition of the VCG mechanism, $\sum_{j \neq i} d_j(Y_j) \geq \sum_{j \neq i} d_j(X_j)$. We conclude that payments are non-negative. Also, since X_1, \dots, X_n maximizes social welfare, $\sum d_i(X_i) \geq \sum d_i(Y_i)$ and hence the payment of agent i is at most

$d_i(X_i)$. Thus the VCG mechanism is individually rational and has non-negative payments. \square

Affine Maximizers and Max-In-Range Mechanisms

We turn to a generalization of the VCG framework that has been used to great success for some problems that are not computationally tractible. An allocation rule \mathcal{A} is *max-in-range* if it always optimizes over its range. That is, \mathcal{A} is specified by some range \mathcal{R} of permitted allocations, and for each input \mathbf{d} returns the optimal partition $\mathbf{x} \in \mathcal{R}$. We claim that such an algorithm \mathcal{A} must be incentive compatible. The proof of this is surprisingly simple: we consider an alternative auction problem in which only partitions from \mathcal{R} are allowed. Then \mathcal{A} can be thought as the optimal solution to this alternative problem, so Theorem 2.3.1 implies that it is incentive compatible.

In fact, the above result is true for the slightly more general class of *affine maximizers*. An affine maximizer is any allocation rule that maximizes the *weighted* social welfare over its range, for some choice of weights. Every affine maximizer is incentive compatible, using a VCG mechanism. A classical result by Roberts [81] (see also Chapter 9 of [77]) is that, in a completely general setting where agents have no restrictions on their utility functions, affine maximizers are the *only* incentive compatible allocation rules.

Theorem 2.3.2 [81] *If there are at least 3 possible outcomes, and players' valuations are unrestricted functions of the outcome, then any deterministic ex post incentive compatible allocation rule is an affine maximizer.*

We note that Roberts' Theorem does not apply to many mechanism design problems of interest, such as combinatorial auctions, since we usually apply heavy restrictions on agent valuations: for instance, we often assume that an agent's value depends only on his own allocation and not the allocations given to other agents.

2.4 Equilibria and Price of Anarchy

If a mechanism is not truthful, then we can expect rational agents to strategically misreport their types in order to maximize their own utilities. Recall that a deterministic bidding strategy for agent i is a function $b_i: V_i \rightarrow V_i$, where we think of $b_i(v_i)$ as the declaration suggested by strategy b_i when the agent's true type is v_i . Given mechanism \mathcal{M} , a *pure Nash Equilibrium* for \mathcal{M} is a profile of bidding strategies \mathbf{b} such that, for all i , all type profiles \mathbf{v} , and all possible declarations d_i for agent i ,

$$u_i(b_i(v_i), \mathbf{b}_{-i}(\mathbf{v}_{-i})) \geq u_i(d_i, \mathbf{b}_{-i}(\mathbf{v}_{-i})).$$

That is, given that all other agents follow strategy profile \mathbf{b} , agent i will maximize his utility by following his strategy b_i . Note that, given a fixed type profile \mathbf{v} , we can equivalently think of a pure Nash Equilibrium as a single profile of type declarations \mathbf{d} , representing the application of strategy \mathbf{b} to types \mathbf{v} .

Agent strategies need not be deterministic. Write $\Delta(S)$ for the space of probability distributions over set S . A randomized bidding strategy for agent i is then a function $b_i: V_i \rightarrow \Delta(V_i)$, where we view $b_i(v_i)$ as a random variable representing a type to declare given true type v_i . A (mixed) Nash Equilibrium for mechanism \mathcal{M} is a profile of randomized strategies in which each agent i maximizes his expected utility by applying his strategy. That is, for all i , all \mathbf{v} , and all d_i ,

$$\mathbf{E}[u_i(b_i(v_i), \mathbf{b}_{-i}(\mathbf{v}_{-i}))] \geq \mathbf{E}[u_i(d_i, \mathbf{b}_{-i}(\mathbf{v}_{-i}))]$$

where the expectations are taken with respect to the randomness in strategy profile \mathbf{b} . Note that we need compare only against a single declaration d_i , rather than a distribution over declarations, since a randomized bidding strategy is optimal if and only if each declaration in its support is optimal.

The (pure, mixed) *Price of Anarchy* for a mechanism $\mathcal{M} = (\mathcal{A}, \mathcal{P})$ is the worst-case ratio, over all agent types and all (pure, mixed) Nash equilibria \mathbf{b} , between the optimal

social welfare and the welfare obtained at strategy \mathbf{b} . That is,

$$PoA_{pure}(\mathcal{M}) = \max_{\mathbf{v}} \max_{\mathbf{b}} \frac{SW_{OPT}(\mathbf{v})}{SW(\mathcal{A}(\mathbf{b}(\mathbf{v})), \mathbf{v})}$$

$$PoA_{mixed}(\mathcal{M}) = \max_{\mathbf{v}} \max_{\mathbf{b}} \frac{SW_{OPT}(\mathbf{v})}{\mathbf{E}_{\mathbf{b}}[SW(\mathcal{A}(\mathbf{b}(\mathbf{v})), \mathbf{v})]}$$

where the maximum over \mathbf{b} is taken over all strategies that are pure Nash equilibria and mixed Nash equilibria for \mathbf{v} , respectively.

Given mechanism \mathcal{M} and distribution \mathbf{F} over types, a *Bayes-Nash equilibrium* for \mathcal{M} and \mathbf{F} is a profile of (possibly randomized) bidding strategies \mathbf{b} such that, for all i , all types v_i for agent i , and all possible declarations d_i for agent i , we have

$$\mathbf{E}_{\mathbf{v}_{-i}}[u_i(b_i(v_i), \mathbf{b}_{-i}(\mathbf{v}_{-i}))] \geq \mathbf{E}_{\mathbf{v}_{-i}}[u_i(d_i, \mathbf{b}_{-i}(\mathbf{v}_{-i}))].$$

That is, given that all other agents' values are drawn from \mathbf{F} and they follow strategy profile \mathbf{b} , agent i will maximize his utility by following strategy b_i . Note that the inequality must hold for every possible type of agent i : the interpretation is that agent i first becomes aware of his type, and then decides whether to follow strategy b_i . For a Bayes-Nash equilibrium, it will be in the agent's best interest to follow strategy b_i for *every* possible type he may have. We note that, by definition, a mechanism is Bayesian incentive compatible if and only if the truthtelling strategy $\mathbf{b}(\mathbf{v}) = \mathbf{v}$ forms a Bayes-Nash equilibrium.

Given distribution \mathbf{F} , the *Bayesian Price of Anarchy* for a mechanism $\mathcal{M} = (\mathcal{A}, \mathcal{P})$ is the worst-case ratio between the expected optimal social welfare and the expected welfare obtained at a Bayes-Nash equilibrium. That is,

$$BPoA(\mathcal{M}) = \max_{\mathbf{b}} \frac{\mathbf{E}_{\mathbf{v}}[SW_{OPT}(\mathbf{v})]}{\mathbf{E}_{\mathbf{v}}[SW(\mathcal{A}(\mathbf{b}(\mathbf{v})), \mathbf{v})]}$$

where the maximum over \mathbf{b} is taken over all strategies that are Bayes-Nash equilibria for \mathbf{F} and \mathcal{M} . Note that the Bayesian price of anarchy is always at least 1.

We can extend the notion of Bayesian price of anarchy to distributions in which types are not independent. For some arbitrary distribution \mathbf{F} over the space of valuation

profiles, a (correlated) Bayes-Nash equilibrium is a profile of bidding strategies such that, for all i , v_i , and d_i ,

$$\mathbf{E}_{\mathbf{v}_{-i}|v_i}[u_i(b_i(v_i), \mathbf{b}_{-i}(\mathbf{v}_{-i}))] \geq \mathbf{E}_{\mathbf{v}_{-i}|v_i}[u_i(d_i, \mathbf{b}_{-i}(\mathbf{v}_{-i}))].$$

The *correlated Bayesian Price of Anarchy* for mechanism \mathcal{M} is then defined precisely as the Bayesian price of anarchy above, where the maximum is taken over strategies in correlated Bayes-Nash equilibrium.

We can also extend equilibria to include approximate utility-maximizing strategies. Given $\gamma \geq 1$, we say that \mathbf{b} is an γ -*approximate pure Nash equilibrium* if no agent can improve his utility by more than a factor of γ by modifying his strategy, given that the other agents follow strategy profile \mathbf{b} . That is, for all i , \mathbf{v} and d_i ,

$$u_i(b_i(v_i), \mathbf{b}_{-i}(\mathbf{v}_{-i})) \geq \frac{1}{\gamma} u_i(d_i, \mathbf{b}_{-i}(\mathbf{v}_{-i})).$$

Note that a 1-approximate Nash equilibrium is precisely a Nash equilibrium. We extend this definition to γ -approximate mixed Nash equilibria and Bayes-Nash equilibria in the natural way.

2.5 Single-Parameter Mechanism Design

In a single-parameter mechanism design problem, each outcome for an agent i can be expressed as a single real value. That is, $O_i = \mathbb{R}$ for each i . An agent's valuation function is assumed to be linear and normalized so that the value of allocation 0 is 0. Each agent's private type can therefore be expressed as a single real number $v_i \in \mathbb{R}$, and the value obtained by agent i for outcome x_i is simply $v_i x_i$.

In Bayesian settings, it is useful to consider agent i 's expected payment and allocation conditioned on his value. To this end, we write $p_i(v_i) = \mathbf{E}_{\mathbf{v}, \mathcal{A}}[p_i(\mathbf{v}) \mid v_i]$ and $x_i(v_i) = \mathbf{E}_{\mathbf{v}, \mathcal{A}}[x_i(\mathbf{v}) \mid v_i]$. The following theorem due to Myerson then characterizes BIC mechanisms.

Theorem 2.5.1 [72] *In a single-parameter setting, a mechanism with allocation rule $\mathbf{x}(\cdot)$ and payment rule $\mathbf{p}(\cdot)$ is BIC if and only if for all agents i :*

- $x_i(v_i)$ is monotone non-decreasing, and
- $p_i(v_i) = v_i x_i(v_i) - \int_0^{v_i} x_i(z) dz + p_i(0)$.

Usually, $p_i(0)$ is assumed to be zero.

We note that the above characterization is independent of the goal of the mechanism designer, be it welfare maximization, makespan minimization, or otherwise. Theorem 2.5.1 motivates the following definition:

Definition 2.5.2 *An allocation rule $\mathbf{x}(\cdot)$ is monotone for distribution \mathbf{F} if $x_i(v_i)$ is monotone non-decreasing for all i . An algorithm is monotone if its allocation rule is monotone.*

Recall that we say an allocation algorithm \mathcal{A} with allocation rule $\mathbf{x}(\cdot)$ is BIC if and only if there exists a payment rule $\mathbf{p}(\cdot)$ such that the mechanism $(\mathbf{x}(\cdot), \mathbf{p}(\cdot))$ is BIC. From Theorem 2.5.1, an algorithm is BIC if and only if it is monotone.

This result has an immediate corollary for ex post incentive compatibility, which was also noted explicitly by Archer and Tardos [4].

Theorem 2.5.3 *A mechanism with allocation rule $\mathbf{x}(\cdot)$ and payment rule $\mathbf{p}(\cdot)$ is ex post incentive compatible if and only if for all agents i and all \mathbf{v}_{-i} :*

- $x_i(v_i, \mathbf{v}_{-i})$ is monotone non-decreasing as a function of v_i , and
- $p_i(v_i, \mathbf{v}_{-i}) = v_i x_i(v_i, \mathbf{v}_{-i}) - \int_0^{v_i} x_i(z, \mathbf{v}_{-i}) dz + p_i(0, \mathbf{v}_{-i})$.

Again, usually, $p_i(0, \mathbf{v}_{-i})$ is assumed to be zero for all i and all \mathbf{v}_{-i} .

We will describe a few examples of single-parameter mechanism design problems studied in the algorithmic game theory literature.

Single-Parameter Combinatorial Auctions The single-parameter combinatorial auction problem is a social welfare maximization problem subject to a feasibility constraint of a particular form. In this problem, there is a base set M of objects and each agent has some (publicly-known) desired set $S_i \subseteq M$. The mechanism chooses a subset of agents to whom to allocate their desired sets, subject to the constraint that no object can be allocated to multiple bidders. That is, a feasible allocation is a binary vector $\mathbf{x} \in \{0, 1\}^n$ such that for all $i \neq j$, $x_i = x_j = 1$ implies $S_i \cap S_j = \emptyset$. Note that this is not the same as the single-minded combinatorial auction problem, where the sets S_i are private information [64].

Related Machine Scheduling for Makespan In parallel related machine scheduling, there is a set of m jobs with publicly-known processing lengths ℓ_1, \dots, ℓ_m . The n agents are thought of as machines, and each machine has a private value s_i that is interpreted as the speed of the machine. The mechanism must generate an assignment that maps each job to a machine; the finishing time of machine i is then the total load of the machine divided by its speed, where the total load is the sum of the lengths of all its assigned jobs. The utility of machine i is taken to be the negative of its finishing time. The interpretation is that machines prefer to be assigned less work.

Note that this problem fits into our framework by setting the private value v_i of each machine to be $-1/s_i$, and allocation \mathbf{x} represents a vector of total loads assigned to each machine under the constraint that it corresponds to a valid assignment of jobs. We note that this feasibility constraint is not downward closed.

Multicast Auctions The multicast auction problem, due to Feigenbaum et al. [39], is described by a graph G with edge weights and a specified root node r . Each agent is associated with a vertex in the graph (where not every vertex need be associated with an agent). The mechanism must generate a connected subgraph H of G that includes vertex r ; the interpretation is that H is a network constructed to connect agents to the

root node. Each agent has a private value for being connected to the root, so that agent i obtains value v_i if his associated node appears in H , otherwise he obtains value 0. Each network H also has a cost, which is taken to be the sum of the weights of the edges in H . The job of the mechanism is to find H that maximizes the sum of the values of the agents connected to r , minus the cost of H .

This problem fits into our framework by taking allocation \mathbf{x} as representing a vector of indicator variables, with $x_i = 1$ indicating that agent i is connected. The cost of \mathbf{x} is then taken to be the cost of the optimal subgraph H that connects the set of agents specified by \mathbf{x} .

2.6 Combinatorial Allocation Problems

A *combinatorial allocation problem* is a mechanism design problem in which a set M of m objects is to be allocated among the participants. An outcome for each agent is a subset of the objects, so that $O_i = 2^M$ for all i . Each agent's type is a valuation function $v_i : 2^M \rightarrow \mathbb{R}$. The goal in this problem is to maximize social welfare. We assume each v_i is monotone and normalized so that $v_i(\emptyset) = 0$. We also tend to assume that the feasibility constraint on valid allocations is *downward-closed*, which in this setting means that if \mathbf{x} is a feasible allocation, then for all i the allocation $(\emptyset, \mathbf{x}_{-i})$ is also feasible.³ We note that, in general, it is not necessarily a requirement that the allocations to the agents be disjoint; the feasibility constraint may allow an object to be allocated to more than one agent simultaneously.

A valuation function v is *single-minded* if there exists a set $S \subseteq M$ and a value $x \geq 0$ such that, for all $T \subseteq M$, $v(T) = x$ if $S \subseteq T$ and 0 otherwise. A valuation function v is *k-minded* if it is the maximum of k single-minded functions. That is, there exist k

³This property is also known in the literature as *separability*. A stronger alternative condition, which is sometimes also referred to as downward-closure but is specific to combinatorial auction problems, requires that if \mathbf{x} is feasible then so is (x'_i, \mathbf{x}_{-i}) for any $x'_i \subseteq x_i$.

sets S_1, \dots, S_k such that for all subsets $T \subseteq M$ we have $v(T) = \max\{v(S_i) \mid S_i \subseteq T\}$. An *additive* valuation function v is specified by m values $x_1, \dots, x_m \in \mathbb{R}_{\geq 0}$, corresponding to the objects $a_i \in M$ for sale, so that $v(T) = \sum_{a_i \in T} x_i$. A valuation function v is *submodular* if it satisfies $v(T) + v(S) \geq v(S \cup T) + v(S \cap T)$ for all $S, T \subseteq M$.

An important special case is the *multi-unit combinatorial auction*, in which there are $B > 1$ copies of each object in M . In the more specialized *multi-unit auction*, it is assumed that all objects are identical; an agent's value therefore depends only on the number of objects she is assigned.

2.6.1 Critical Prices

A type vector \mathbf{v} and an allocation rule \mathbf{x} for a combinatorial allocation problem defines *critical prices*, $\theta_i(S, \mathbf{v}_{-i})$, for any agent i and set $S \subseteq M$. The value $\theta_i(S, \mathbf{v}_{-i})$ is the minimum amount that agent i could bid on set S and still win S , assuming the other agents bid according to \mathbf{v}_{-i} . That is,

$$\theta_i(S, \mathbf{v}_{-i}) = \inf\{z : \exists d_i \text{ such that } d_i(S) = z \text{ and } x_i(d_i, \mathbf{v}_{-i}) = S\}.$$

We note that this definition of critical prices holds even if it is not the case that increasing one's declared value for a set necessarily increases the probability of obtaining that set. However, most of the mechanisms we consider in this thesis do satisfy this monotonicity property, which motivates the terminology of a "critical" price.

From Bartal, Gonen and Nisan [11], we have the following characterization of truthful mechanisms for combinatorial auctions.

Theorem 2.6.1 *A mechanism \mathcal{M} is dominant strategy truthful if and only if, for every bidder i and every type profile \mathbf{v} ,*

1. $x_i(\mathbf{v}) \in \operatorname{argmax}_S \{v_i(S) - \theta_i(S, \mathbf{v}_{-i})\}$ and
2. $p_i(\mathbf{v}) = \theta_i(x_i(\mathbf{v}), \mathbf{v}_{-i})$.

Theorem 2.6.1 states that a truthful mechanism must charge critical prices as its payments, and must always allocate to each agent a set that maximizes his utility subject to these prices.

2.6.2 Input Representation

We wish to study polynomial-time algorithms for combinatorial allocation problems. We immediately face an input representation problem: we would like for our algorithm to be polynomial in the number of agents n and the number of objects m , but as stated its input is a sequence of valuation functions, whose representation is exponential in m . We must therefore be careful in specifying the representation of our problem.

A standard approach is to specify a *bidding language* with which to describe valuation functions. A full description of the various languages that have appeared in the literature is beyond the scope of this survey; we will introduce one of the simplest. In the *XOR bidding language*, an agent specifies a list of (set,value) pairs; each pair is referred to as a *bid*. A bid (S, v) indicates that the agent would obtain a value of v for any set containing S . These bids are mutually exclusive, meaning that if an agent is given a set that satisfies multiple bids then he obtains the maximum of the values of the satisfied bids. Thus, a list of k bids $\{(S_j, v_j)\}$ corresponds to the k -minded valuation function $v(S) = \max\{v_j : S_j \subseteq S\}$. Under this bidding language, we would require that an algorithm be polynomial in n , m , and k , the maximum number of bids put forward by any agent.

Another approach assumes that valuation functions are presented to an algorithm as black box *oracles*. We think of each agent as presenting a single oracle, representing that agent's valuation function, to the mechanism; the oracle can then be queried to reveal information about that agent's valuation. Each query is counted as a single operation, and the nature of the allowable queries is determined by a *query model*. For example, in the *value query* model, the mechanism performs a query by presenting a set, and

the oracle returns the value for that set. In the *demand query* model, the algorithm performs a query by presenting a vector \mathbf{p} of m prices, one per object in $a_j \in M$, and the oracle returns the set S that would maximize set value minus sum of prices of objects in the set. That is, given price vector \mathbf{p} , the oracle for agent i would return $S \in \operatorname{argmax}_S \{v_i(S) - \sum_{a_j \in S} p_j\}$. By contrast, the general query model allows arbitrary queries to the oracles; it is used primarily for proving lower bounds.

For many of our results we will think of the input to the combinatorial allocation problem as specified by bids for desired sets via the XOR bidding language, though we will point out instances in which oracle models are particularly relevant.

2.6.3 Examples

We now describe a number of particular instances of combinatorial allocation problems, with some relevant approximation algorithms for each.

Combinatorial Auctions The general combinatorial auction problem is defined by the feasibility constraint that no two allocations can intersect. Lehmann et al [64] show that the (non-adaptive) greedy allocation rule that ranks bids according to the ranking function $\frac{v}{\sqrt{|S|}}$ (i.e. for a bid of value v for set S) achieves a $\sqrt{2m}$ approximation ratio for CAs. An alternative allocation rule, which can be implemented with a polynomial number of demand queries, was proposed by Mu'alem and Nisan [71]. This allocation rule combines a standard greedy algorithm with an allocation of all objects to a single bidder.

These approximation factors are tight subject to standard complexity constraints. Even if all agents are assumed to be single-minded, a simple reduction to CLIQUE [84] shows that it is NP-hard to approximate this problem to within a factor of $O(m^{\frac{1}{2}-\epsilon})$.⁴

⁴Note that there are trivial algorithms that obtain linear performance in n , but it is usually assumed that $n \gg \sqrt{m}$.

The best-known deterministic dominant strategy truthful mechanism for the combinatorial auction problem obtains an approximation factor of $O(\frac{m}{\sqrt{\log m}})$ [51]. For randomized mechanisms better results are known: a truthful-in-expectation $O(\sqrt{m})$ -approximation was given by Lavi and Swamy [62], and a universally truthful $O(\sqrt{m})$ -approximate mechanism was later found by Dobzinski, Nisan, and Schapira [33].

Cardinality-restricted Combinatorial Auctions In the special case that players' desires are restricted to sets of size at most k , the non-adaptive greedy algorithm with $r(i, S, v) = v$ is k -approximate assuming single-minded agents. This translates to a $(k + 1)$ approximate algorithm for general agents. As with the general combinatorial auction problem, the best-known deterministic dominant strategy truthful mechanism for this problem has approximation factor $O(\frac{m}{\sqrt{\log m}})$ [51].

Multiple-Demand Unsplittable Flow Problem In the unsplittable flow problem (UFP), we are given an undirected graph with edge capacities. The objects are the edges, and each valuation function is such that agent i has some value $v(s, t)$ for being given a path from s to t . Each agent additionally specifies a fractional demand $d_i \in [0, 1]$ corresponding to a desired amount of flow to send along the given path. An allocation is feasible if the total allocated flow along each edge is no more than its capacity. Let B be the minimum edge capacity. A primal-dual algorithm, which is an adaptive greedy allocation rule, obtains an $O(m^{1/(B-1)})$ approximation for any $B > 1$ [18].

Convex Bundle Auctions In a convex bundle auction, M is the plane \mathbb{R}^2 , and allocations must be non-intersecting compact convex sets. We suppose that agents declare valuation functions by making bids for such sets. Given such a collection of bids, the aspect-ratio, R , is defined to be the maximum diameter of a set divided by the minimum width of a set. A non-adaptive greedy allocation rule using a geometrically-motivated priority function yields an $O(R^{4/3})$ approximation [7]. Alternative greedy algorithms

yield better approximation ratios for special cases, such as rectangles.

Max-profit Unit Job Scheduling In this problem, each bidder has a job of unit time to schedule on one of multiple machines. A bidder has various windows of time of the form (release time, deadline, machine) in which his job could be scheduled, with a potentially different profit resulting from each window. The profits and windows are private information to each bidder. The goal of the mechanism is to schedule the jobs to maximize the total profit. The standard greedy algorithm obtains a 3-approximation [70].

Unlike the previous examples, an exact algorithm is known for the case of single-minded bidders [9], which uses dynamic programming and runs in time $O(n^7)$. The VCG mechanism can therefore be implemented using this algorithm, leading to a dominant strategy truthful mechanism that optimizes social welfare.

2.7 Related Work

Deterministic Truthful Mechanisms The research agenda of constructing truthful, computationally efficient algorithms was first proposed by Nisan and Ronen [76], and since that time the problem of developing ex post incentive compatible mechanisms for social welfare problems has been very well studied. This pursuit has been particularly successful for social welfare maximization in single-parameter problems, where numerous deterministic truthful mechanisms are known. Lehmann et al. [64] provided a truthful $O(\sqrt{m})$ -approximate greedy mechanism for the single-minded combinatorial auction problem, improved to an $\epsilon\sqrt{m}$ -approximation by Mu'alem and Nisan [71]. This matches the lower bound of $\Omega(m^{\frac{1}{2}-\epsilon})$, which was established by Nisan via communication complexity [75] and also by Sandholm for bidders with succinctly-representable valuations, assuming $P \neq NP$, via a reduction to CLIQUE [84, 50]. Briest et al. [18] described a general method for transforming FPTAS algorithms for single-parameter problems into

truthful FPTAS algorithms. This yields a truthful FPTAS for the single-minded multi-unit auction problem, improving upon the previously best-known 2-approximation [71]. Briest et al. also present a general method for constructing truthful primal-dual algorithms for single-minded packing problems such as multi-unit combinatorial auction problems. Babaioff and Blumrosen [7] give truthful greedy algorithms for single-minded auctions of convex bundles in the plane, such as auctions for advertising space in printed media. Chekuri and Gamzu described a method for constructing truthful mechanisms via an iterative greedy approach, and use this to obtain a $(2 + \epsilon)$ approximation for the single-parameter multiple knapsack problem [21].

The design of deterministic truthful algorithms appears more difficult in general multi-parameter settings. The best-known truthful deterministic mechanism for the general CA problem proceeds by dividing the objects arbitrarily into $O(\log m)$ equal-sized indivisible bundles, then allocating those bundles optimally; this achieves an approximation ratio of $O(m/\sqrt{\log m})$ [51]. Due to the lower bound of $\Omega(m^{\frac{1}{2}-\epsilon})$ for this problem, discussed above, much work has been dedicated to special cases in which valuation functions can be assumed to be of a particular form. For the case of submodular bidders, the best-known deterministic truthful auction obtains an $O(\sqrt{m})$ approximation [32], whereas straightforward algorithms obtain constant-factor approximations [63, 58]. Dobzinski and Nisan [31] construct a truthful 2-approximate mechanism for multi-unit auctions (ie. having many copies of just a single object), and a truthful PTAS when additionally each declaration can be represented as the maximum of k single-minded desires. All of the above constructions are examples of max-in-range mechanisms. For the case of multi-unit combinatorial auctions, when there are $B \geq 3$ copies of each object, Bartal et al. [11] give a greedy algorithm that obtains an $O(Bm^{\frac{1}{B-2}})$ approximation.

The apparent gaps between the power of truthful and non-truthful algorithms for multi-parameter problems spawned a significant line of research aimed at giving lower bounds on the approximating power of deterministic truthful algorithms. Papadimitriou,

Schapira and Singer [80] gave an example of a social welfare problem for which constant-factor approximation algorithms exist, but any polytime deterministic ex post incentive compatible mechanism attains at best a polynomial approximation factor. A similar gap for the submodular combinatorial auction problem was established by Dobzinski [29]. Lehmann, Mu’alem, and Nisan [60] show that any truthful combinatorial auction mechanism that uses a suitable bidding language, is unanimity-respecting, and satisfies an independence of irrelevant alternatives property (IIA) cannot attain a polynomial approximation ratio. It has also been shown that, roughly speaking, any truthful polytime subadditive combinatorial auction mechanism with an approximation factor better than 2 cannot satisfy the natural property of being *stable*⁵ [34]. Dobzinski and Nisan showed that no max-in-range algorithm for the combinatorial auction problem can obtain an approximation ratio better than $\Omega(\sqrt{m})$ with polynomial communication between agents and the mechanism [30]. This was later extended to show that no max-in-range algorithm can obtain an approximation ratio better than $\Omega(\sqrt{m})$ even when agents have succinctly-representable valuations (i.e., budget-constrained additive valuations) [19].

Randomized Truthful Mechanisms While deterministic truthful mechanisms appear limited for multi-parameter problems, randomized mechanisms appear much more promising. Lavi and Swamy [62] consider mechanisms for multi-parameter packing problems and show how to design a β -approximation mechanism that is truthful in expectation from any β -approximation that verifies an integrality gap. This implies an $O(\sqrt{m})$ truthful in expectation mechanism for the general combinatorial auction problem, which was later improved Dobzinski, Nisan and Schapira [33] to an $O(\sqrt{m})$ mechanism that is universally truthful. Dughmi, Roughgarden and Yan [37] extend the notion of designing mechanisms based upon randomized rounding algorithms, and obtain truthful in expectation mechanisms for a broad subclass of submodular combinatorial auctions. For the class

⁵In a stable mechanism, no player can alter the outcome (i.e. by changing his declaration) without causing his own allocated set to change.

of general submodular combinatorial auctions, a universally truthful $O(\log m \log \log m)$ approximation was given by Dobzinski [28]. Dughmi and Roughgarden [36] give a construction that converts any FPTAS algorithm for a social welfare problem into a mechanism that is truthful in expectation, by way of a variation on smoothed analysis. In the single-parameter setting, Archer et al. [3] considered multi-unit combinatorial auctions where there are many (at least logarithmic) copies of each item and gave a $(1 + \epsilon)$ -approximation mechanism.

Given the above results, it is tempting to conjecture that truthful-in-expectation mechanisms match the power of arbitrary algorithms for approximating social welfare. However, Dobzinski [29] gives an example of a social welfare problem for which constant-factor approximation algorithms exist, but any polytime truthful in expectation mechanism that uses value queries attains at best a polynomial approximation factor.

For the goal of approximating the makespan for related machine scheduling, Tardos [4] gave a (single-parameter) related machine scheduling mechanism that yields a 3-approximation to the makespan. This approximation factor was improved over the course of numerous works, culminating in a randomized polynomial-time approximation scheme (PTAS) that is truthful in expectation due to Dhangwatnotai et al. [27] and a deterministic truthful PTAS due to Christodoulou and Kovacs [22].

Mechanisms at equilibrium The notion of Bayes-Nash equilibrium as a model of rationality under partial information was introduced by Harsanyi [46]. For an overview of the development and impact of this theory we recommend a review by Myerson [73]. This and other equilibrium notions are common solution concepts for mechanism design in the economic literature; see Jackson [55] for a survey.

The inefficiency of equilibria is well-studied in the computational game theory literature, wherein the worst-case approximation ratio at equilibrium is referred to as the *price of anarchy* (introduced by Papadimitriou [79]). Inefficiency of equilibria is most

commonly studied in settings in which agents choose their outcomes directly (eg. routing games [83]) rather than through a mechanism. The literature includes many refinements of these concepts, such as convergence of potential games and price of total anarchy. See chapters 17-21 of [77] and references therein.

The BNE solution concept has recently been applied to submodular combinatorial auctions [23], where it was shown that a randomized mechanism can attain a 2-approximation at any mixed equilibrium assuming that bidders are ex-post individually rational. Subsequent to the work in this thesis, a similar auction method due to Bhawalkar and Roughgarden obtains a 2-approximation at equilibrium and a $2 \log m$ -approximation at Bayes-Nash equilibrium for subadditive bidders [13]. Gairing et al. [40] consider Bayes-Nash equilibria of a routing game and study worst-case performance. Paes Leme and Tardos [65] studied the performance of the generalized second price auction for advertising slots at equilibrium; this analysis was then extended by Lucier and Paes Leme [78] and Caragiannis et al [20]. Pure equilibria of first-price mechanisms have also been studied for path procurement auctions [53]. The problem of designing auctions that maximize revenue at Nash equilibrium has been extensively studied; notably in work on Internet advertising slot auctions [85, 38].

There are various alternatives to the single-shot equilibrium concept in the literature, many of which are based upon mechanisms that proceed in rounds. The study of regret-minimization goes back to the work of Hannan on repeated two-player games [45]. Kalai and Vempala [57] extend the work of Hannan to online optimization problems, and Kakade et al [56] further extend to settings of approximate regret minimization. Blum et al [14] apply regret-minimization to the study of inefficiency in repeated games, coining the phrase “price of total anarchy” for the worst-case ratio between the optimal objective value and the average objective value when agents minimize regret. Properties of best-response dynamics in repeated games, and especially the question of convergence to a pure equilibrium, is well-studied (see Chapter 19 of [77]). The study of average

performance of best-response dynamics as a metric of game inefficiency, the so-called “price of sinking,” was introduced by Goemans et al [41]. Babaioff et al. [8] look at the equilibrium notion of *algorithmic implementation in undominated strategies* and give a technique for turning a β -algorithm into a $\beta(\log v_{max})$ -approximation mechanism. This solution concept requires that no agent plays a strategy that is dominated by an easy to find strategy. Their approach applies to single-valued combinatorial auctions and does not require the mechanism to know which bundles each agent desires. The mechanism they present implements such auctions in a multi-round fashion.

Greedy Algorithms Some of our results make crucial use of the nature of greedy allocation algorithms. Properties of greedy algorithms have been extensively studied. Borodin et al [17] introduced the notion of priority algorithms as a model for greedy algorithms, and studied their power in solving various approximation problems. Monotone greedy algorithms for combinatorial auctions were studied first by Lehmann et al [64], then subsequently by Mu’alem and Nisan [71] and Briest, Krysta, and Vocking [18], resulting in the development of new incentive compatible algorithms for single-minded bidders. Gonen and Lehmann [42] showed that no algorithm that greedily accepts bids for sets can guarantee an approximation better than \sqrt{m} for the general CA problem. More generally, Krysta [59] showed that no oblivious greedy algorithm (in our terminology: fixed order greedy priority algorithm) obtains approximation ratio better than \sqrt{m} . This contrasts with our results in Chapter 7 in that we consider the even more general class of all priority algorithms but restrict them to be incentive compatible.

The class of priority algorithms is loosely related to the notion of online algorithms. Mechanism design has been studied in a number of online settings, and lower bounds are known for the performance of truthful algorithms in these settings [61, 68]. The critical difference between these results and our results for priority algorithms is that a priority algorithm has control over the order in which input items are considered, whereas in an

online setting this order is chosen adversarily.

Part I

Single-Parameter Problems

Chapter 3

Single-Parameter Bayesian Incentive Compatibility

We begin our exploration of Bayesian algorithmic mechanism design in the realm of single-parameter problems. In such a setting, we view the mechanism as providing a service to a collection of agents, where each agent has a single independent private value for receiving service. The mechanism must decide upon a level of service for each agent, subject to feasibility and/or cost constraints, with the goal of maximizing social welfare.

In the arena of mechanism design, the above optimization must be performed under the simultaneous restrictions imposed by agent incentives and computational feasibility. To address the former, we will require that our solution be Bayesian incentive compatible (BIC). The latter issue imposes the constraint that our mechanism must execute in polynomial time. Thus, for some computationally infeasible problems, we must content ourselves with mechanisms that approximately optimize social welfare. Moreover, we cannot expect our mechanisms to beat the approximations attainable by algorithms that are not constrained by agent incentives. The best result we can hope for, then, is a Bayesian incentive compatible mechanism matching the performance of the best-possible

The results in this chapter are based upon joint work with Jason Hartline [48].

approximation algorithm, for every problem.

In this chapter we establish such a result by way of a transformation that converts any approximation algorithm for a single-parameter social welfare problem into a computationally efficient BIC mechanism without loss of performance. In other words, in settings of partial information, we can reduce the problem of truthful mechanism design to the problem of algorithm design.

Agent incentives in Bayesian mechanism design are very well understood in single-parameter settings. Recall from Section 2.5 that a mechanism is Bayesian incentive compatible if and only if (a) the probability an agent is served is monotone non-decreasing in the agent's private value, and (b) the agent's expected payment is of a particular form identified from the allocation rule. The task of designing a BIC mechanism therefore reduces to the problem of designing an algorithm with monotone allocation rules (i.e. a BIC algorithm).

The main challenge in reducing incentive compatible mechanism design to algorithm design is that approximation algorithms do not generally have monotone allocation rules. Our reduction shows that in a Bayesian setting we can convert any non-monotone allocation rule into a monotone one without compromising its social welfare. The main technical observation that enables this reduction is that, in a Bayesian setting, we can focus on a single agent for whom the allocation rule is not monotone, apply a transformation to his declared value that fixes this non-monotonicity (and weakly improves our objective), and *no other agents are affected* (in a Bayesian sense). Therefore, we can apply the transformation independently to each agent.

Our reduction is as follows:

1. For each agent i , generate a transformation $\pi_i: V_i \rightarrow V_i$. This transformation depends on the distribution and algorithm and can be constructed prior to considering any agent bids.
2. For each agent i , apply π_i to the declared input value v_i .

3. Run the approximation algorithm on the resulting bids, $\pi(\mathbf{v})$, and output its solution.

We will build our transformations in such a way that distribution F_i is stationary under π_i , for each i . Notice, then, that the application of these transformations does not alter the prior distribution of values.

At the heart of our construction lies the definition of π_i in step 1 of the reduction. To build transformations that guarantee incentive compatibility without degrading social welfare or modifying priors, we develop a monotoning technique for allocation rules (adapted from the standard *ironing procedure* from the field of Bayesian optimal mechanism design [72]). Assuming that we have functional access to the prior distributions and the allocation rules of the original algorithm, and that we can apply arbitrary calculus on those functions, we can construct transformations with the desired properties and therefore convert the given approximation algorithm into a BIC mechanism. We describe the overall nature of this transformation in Section 3.2.

In many cases, it may be unreasonable to assume that we have access to the functional form of the allocation rule. Moreover, even if we are given access to the allocation rule, it is not clear that the necessary calculus can be applied in polynomial time. To address these issues, we modify our reduction to apply in a more restricted computational model, where we have only black-box query access to the original algorithm and sampling access to the prior distribution. Our reduction can proceed in such a restricted model with only a slight change: we must estimate the allocation rule by sampling the distribution and making black-box calls to the algorithm. These estimates can be made precise enough to enable arbitrary small loss in welfare. We arrive at the following result (stated formally in Section 3.3):

Theorem 3.3.1 *For any $\epsilon > 0$ and any single-parameter social welfare problem, a Bayesian incentive compatible algorithm \mathcal{A}' can be computed from any algorithm \mathcal{A} . Its expected social welfare satisfies $\mathcal{A}' \geq \mathcal{A} - \epsilon$, and its runtime is polynomial in n , $1/\epsilon$, and*

the runtime of \mathcal{A} .

That is, for any ϵ , we give a black box reduction that, in polynomial time in the number of agents and $1/\epsilon$, converts any approximation algorithm into a BIC algorithm with an additive loss of ϵ to the social welfare. We also give a pseudo-polynomial time reduction to a BIC algorithm with a multiplicative loss of ϵ , and a fully polynomial time approximation scheme under some mild conditions on the original algorithm. For this last result, what we require is that the expected social welfare of the original algorithm not be more than polynomially smaller than the largest expected value of any single agent. This is trivially satisfied for many cases of interest, such as downward-closed feasibility problems.

We conclude that there is no gap between algorithmic approximation and approximation by Bayesian incentive compatible mechanisms. We note, however, that the same question can be asked for other non-welfare-maximization objectives. We show that for some objectives such a gap is essential, but for other natural objectives (such as makespan minimization) the question of a general reduction remains open. We do demonstrate that the methodology outlined in this chapter cannot yield a general reduction for makespan, so any such reduction necessarily entails a significant deviation from our approach.

We review some facts about single-parameter mechanism design in Section 3.1. In Section 3.2 we describe the form of our reduction, under an ideal model with arbitrary functional access to the prior distribution and original algorithm. Then in Section 3.3 we give the details of our main result: a general black-box reduction with additive error, or multiplicative error under some mild assumptions on the original algorithm. Finally, in Section 3.4 we highlight some limitations of our approach: it does not preserve worst-case approximation factors, and cannot be applied to other optimization objectives such as makespan minimization.

Following the original publication of this work, others have extended the results in this chapter to multi-parameter settings [12, 47]. In the multi-parameter setting, there

are problems for which Bayesian incentive compatible mechanisms achieve approximation factors that are not possible for dominant strategy incentive compatible mechanisms [80]. These separation results highlight the importance of the Bayesian setting, as it allows one to address important mechanism design problems that are infeasible with dominant strategy truthfulness.

3.1 Preliminaries

Our results focus upon binary single-parameter social welfare problems. An algorithm in such a setting must select a set of agents to serve. This *allocation* is $\mathbf{x} = (x_1, \dots, x_n)$ where $x_i \in \{0, 1\}$ is an indicator for whether or not agent i is served. Agent i has *valuation* $v_i \in \mathbb{R}_{\geq 0}$ for being served, so that the value obtained by agent i for outcome \mathbf{x} is simply $v_i x_i$.

This class of problems includes single-parameter combinatorial auctions, facility location problems, and many others. In the most general form, we assume that the auctioneer has an arbitrary cost function $c(\cdot)$ for generating various allocations and wishes to maximize social welfare. That is, we wish to choose \mathbf{x} in order to maximize the social welfare function $\mathbf{v} \cdot \mathbf{x} - c(\mathbf{x})$ for some arbitrary function $c(\cdot)$. Note that costs can be arbitrarily non-monotone or the set system non-downward-closed (e.g. public good or scheduling problems). The *general feasibility setting* is the special case where costs are zero (feasible) or infinity (infeasible). An important subclass are *downward-closed settings* where any subset of a feasible set is feasible. Downward closed settings capture the class of packing problems, and include single-minded combinatorial auctions [64] and knapsack auctions [1].

Recall from Section 2.5 that given a mechanism $\mathcal{M} = (\mathcal{A}, \mathcal{P})$, where \mathcal{A} is an allocation algorithm with allocation rule \mathbf{x} and \mathcal{P} is a payment algorithm with payment rule \mathbf{p} , we write $p_i(v_i) = \mathbf{E}_{\mathbf{v}, \mathcal{P}}[p_i(\mathbf{v})|v_i]$ and $x_i(v_i) = \mathbf{E}_{\mathbf{v}, \mathcal{A}}[x_i(\mathbf{v})|v_i]$ for the expected payment and

allocation to agent i , given that agent i reports value v_i . Further recall that an algorithm \mathcal{A} is Bayesian incentive compatible (BIC) precisely if $x_i(\cdot)$ is a monotone non-decreasing function for each i . Finally, recall that we will write \mathcal{A} for the expected social welfare generated by allocation algorithm \mathcal{A} under implicit distribution \mathbf{F} . That is, we write $\mathcal{A} = \mathbf{E}_{\mathbf{v} \sim \mathbf{F}}[SW(\mathcal{A}(\mathbf{v}), \mathbf{v})]$.

Without loss for non-negative bounded-support distributions, we will assume $v_i \in [0, 1]$.¹ A valuation profile is thus a vector $\mathbf{v} = (v_1, \dots, v_n) \in [0, 1]^n$. A mechanism for such a problem is then specified by payment rules $p_i: [0, 1] \rightarrow \mathbb{R}_{\geq 0}$ and allocation rules $x_i: [0, 1] \rightarrow [0, 1]$.

3.2 The Approach

In this section we develop the economic insight that underlies our construction, glossing over certain computational details. To this end, we envision an *ideal model*, in which we are given access to the functional form of the distribution \mathbf{F} and allocation rules \mathbf{x} of an algorithm, and can perform arbitrary calculus on those functions when designing our mechanism. We discuss the computational issues hidden by this idealized model at the end of the section.

In the ideal model we prove that, for any single-parameter social welfare problem and any polytime algorithm, there exists a polytime BIC mechanism with the same expected social welfare.

Theorem 3.2.1 *In the ideal model, for any algorithm \mathcal{A} , there is a BIC algorithm $\bar{\mathcal{A}}$ such that $\bar{\mathcal{A}} \geq \mathcal{A}$, where the runtime of $\bar{\mathcal{A}}$ is polynomial in n and the runtime of \mathcal{A} .*

We note that Theorem 3.2.1 applies to randomized algorithms in addition to deterministic algorithms. Also, Theorem 3.2.1 implies an immediate corollary for Bayesian approximation.

¹The bounded support assumption is unnecessary except for our results using sampling.

Corollary 3.2.2 *Given any polytime Bayesian β -approximation algorithm \mathcal{A} , there exists a BIC polytime Bayesian β -approximation $\bar{\mathcal{A}}$.*

Corollary 3.2.2 applies in the special case that \mathcal{A} is a worst-case β -approximation, but the resulting BIC algorithm $\bar{\mathcal{A}}$ will not necessarily be a worst-case β -approximation. An example is given in Section 3.4.1.

Let us build some intuition for the requirements of Theorem 3.2.1. Suppose that we are given an algorithm \mathcal{A} that is monotone for the distribution \mathbf{F} . Then \mathcal{A} is already BIC (by Theorem 2.5.1) and specifies the allocation rule $\mathbf{x}(\cdot)$, so we must only compute the payment rule. This computation is straightforward given the formula from Theorem 2.5.1.

Now suppose we have a non-monotone Bayesian β -approximation algorithm \mathcal{A} with allocation rule $\mathbf{x}(\cdot)$. We would like to use \mathcal{A} to construct a monotone algorithm $\bar{\mathcal{A}}$ from which we can obtain a BIC mechanism by simply computing the payment rule as above. We must make sure that in doing so we do not reduce the algorithm's expected welfare. The key property of our approach which makes it tractable is that we monotinize each agent's allocation rule independently without changing (in a Bayesian sense) the (expected) allocation rule any other agent faces. This property is also important for the approximation factor as \mathcal{A} is guaranteed to be a Bayesian β -approximation only for the given distribution \mathbf{F} , and may not be a good approximation for some other distribution.

In summary, the desiderata for monotonizing agent i are:

- D1. monotone $\bar{x}_i(v_i)$,
- D2. (weakly) improved social welfare $\mathbf{E}_{v_i}[v_i \bar{x}_i(v_i)] \geq \mathbf{E}_{v_i}[v_i x_i(v_i)]$, and
- D3. other agents unaffected in expectation.

Notice that if we satisfy the last condition we can apply the process simultaneously to all agents.

3.2.1 Ironing via Resampling

There is a history of fixing non-monotonicities in Bayesian mechanism design. Myerson invented the technique of *ironing* which relies on the fact that if an allocation rule is constant over some interval then any agent within that interval is effectively equivalent to a canonical “average” agent from that interval. Myerson applied this theory to iron *virtual valuation functions* which are used in Bayesian profit maximization [72]. We will apply this theory directly to allocation rules. We note that while our approach borrows from techniques used by Myerson’s optimal mechanism, the mechanism we construct is fundamentally different; see Section 3.2.4.

Before we describe our ironing procedure in full, let us develop some more intuition. Suppose allocation rule $x_i(\cdot)$ of \mathcal{A} is non-monotone for agent i . A simple approach to flattening non-monotonicities is to choose some interval $[a, b]$ on which $x(\cdot)$ is non-monotone, and to treat the agent identically whenever his value falls within this interval. For example, whenever $v_i \in [a, b]$ we could choose to pretend that v_i is actually some other fixed value v' (e.g. $v' = a$) and pass this “pretend” value v' to the algorithm. Unfortunately, if we take this naïve approach, we would have changed the distribution of agent i ’s input to the algorithm (in particular, the probability of value v' would be increased) and violated D3. In order to maintain D3 we make a minor modification: instead of picking a fixed v' , we will draw v' from F_i restricted to the interval $[a, b]$. Thus, we are replacing $v_i \in [a, b]$ with v' drawn from the same distribution. Other agents cannot tell the difference – this operation does not change the distribution of agent i ’s input! Moreover, agent i will indeed be treated identically whenever $v_i \in [a, b]$: the new probability of allocation will be precisely the distribution weighted average of $x_i(\cdot)$ over the interval $[a, b]$.

Let $x_i'(\cdot)$ represent the allocation rule obtained from the following procedure (where $\mathbf{v}_{-i} \sim \mathbf{F}_{-i}$):

- if $v_i \in [a, b]$, redraw $v' \sim F_i$ restricted to $[a, b]$; else, set $v' = v_i$.

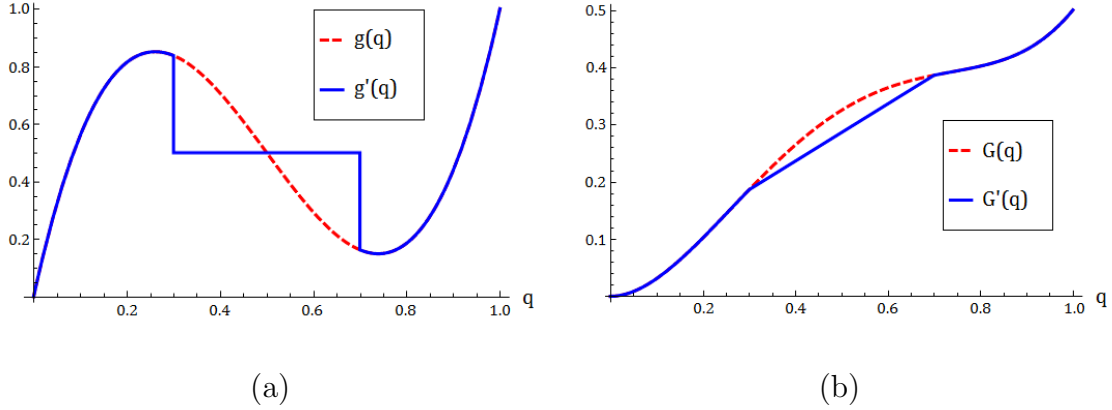


Figure 3.1: (a) A non-monotone ironing g' (solid) of curve g (dashed). (b) The corresponding integral curves G' (solid) and G (dashed) in probability space.

- run $\mathcal{A}(v', \mathbf{v}_{-i})$.

We say that $x_i'(\cdot)$ is the curve $x_i(\cdot)$ *ironed on interval* $[a, b]$. We note that $x_i'(v_i) = x_i(v_i)$ for $v_i \notin [a, b]$ and $x_i'(v_i) = \mathbf{E}_{v' \sim F_i}[x_i(v') \mid v' \in [a, b]]$ otherwise. Indeed, we can express x_i' as $x_i'(v) = x_i(\pi_i(v))$ where π_i is the (randomized) mapping from v_i to v_i' . Note that we can easily iron along multiple disjoint intervals, redrawing v' from whichever interval contains v_i' (if any).

We now explore a method for choosing intervals on which to iron in order to obtain monotonicity. It will be instructive to consider the *allocation rule in probability space* instead of valuation space, and the *cumulative allocation rule* (also in probability space).

- Let $g(q) = x_i(F_i^{-1}(q))$ be the allocation rule in probability space.
- Let $G(q) = \int_0^q g(z)dz$ be the cumulative allocation rule.

Notice that monotonicity of $x_i(\cdot)$ is equivalent to monotonicity of $g(\cdot)$ which is equivalent to convexity of $G(\cdot)$.

Let $x_i'(\cdot)$ be $x_i(\cdot)$ ironed along some interval $[a, b]$, and consider the corresponding curves $g'(\cdot)$ and $G'(\cdot)$. This ironing procedure corresponds to replacing $g(\cdot)$ with its average on $[a, b]$, or equivalently $G(\cdot)$ with the line segment connecting $G(F(a))$ to $G(F(b))$

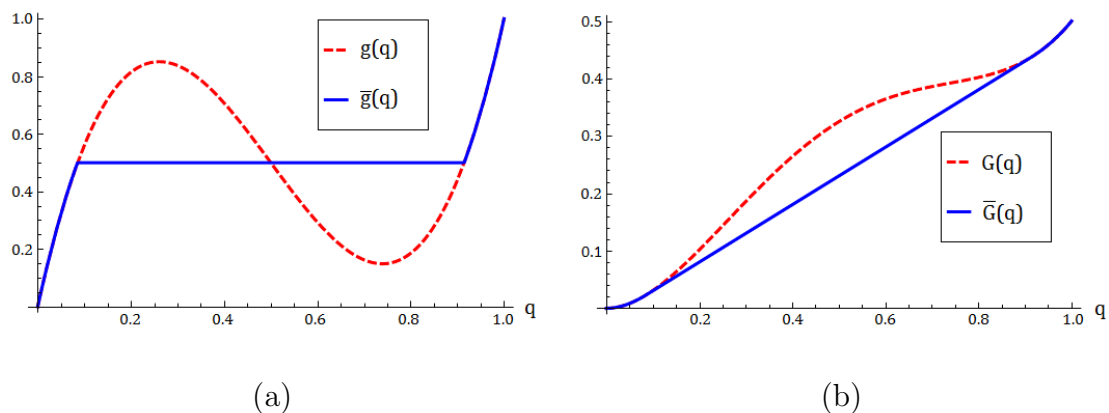


Figure 3.2: (a) A monotone ironing \bar{g} (solid) of curve g (dashed). (b) The corresponding integral curves \bar{G} (solid) and G (dashed) in probability space. Note \bar{G} is the convex hull of G .

(See Figure 3.1).² This latter line segment interpretation suggests that we can view our interval selection problem as the problem of replacing portions of curve G with straight line segments so that the resulting curve \bar{G} will be convex. This is precisely the problem of finding the convex hull of G . Thus the choice of intervals that monotonezes $x_i(\cdot)$ (satisfying D1) is precisely the set of intervals defined by the convex hull of $G(\cdot)$. See Figure 3.2.

Finally, since the convex hull of $G(\cdot)$ lies below $G(\cdot)$, this transformation weakly improves welfare (satisfying D2). Informally speaking, in moving from cumulative allocation rule $G(\cdot)$ to $\bar{G}(\cdot)$, we lower the probability of low-value allocations in exchange for a corresponding increase in the probability that higher-valued allocations occur. This intuition is made more precise in Lemma 3.2.7, below.

3.2.2 The Ironed Algorithm

We are now ready to define our ironed algorithm $\bar{\mathcal{A}}$. Given distribution F and interval I , we will write $F[I]$ to mean F restricted to I .

²Note that the transformation to probability space (from valuation space) is necessary for obtaining this line-segment interpretation.

Definition 3.2.3 ($\text{RESAMPLE}(\mathcal{A}, \mathcal{I})$) *Given algorithm \mathcal{A} and a profile of sets of disjoint intervals, $\mathcal{I} = \{\mathcal{I}_1, \dots, \mathcal{I}_n\}$, the resampled algorithm for \mathcal{A} with intervals \mathcal{I} is algorithm $\text{RESAMPLE}(\mathcal{A}, \mathcal{I})$:*

1. *For each agent i , if $v_i \in I \in \mathcal{I}_i$, draw $\pi_i(v_i) \sim F_i[I]$; else, let $\pi_i(v_i) = v_i$.*
2. *Run $\mathcal{A}(\pi(\mathbf{v}))$.*

Definition 3.2.4 ($\text{MONOINTS}(\mathbf{x})$) *The set of monotoning intervals for $\mathbf{x}(\cdot)$, denoted $\text{MONOINTS}(\mathbf{x})$, is the set of intervals $(\mathcal{I}_1, \dots, \mathcal{I}_n)$ defined by:*

1. *Let $g_i(q) = x_i(F_i^{-1}(q))$ be the allocation rule in probability space.*
2. *Let $G_i(q) = \int_0^q g_i(z) dz$ be the cumulative allocation rule.*
3. *Let $\bar{G}_i(\cdot)$ be the convex hull of $G_i(\cdot)$.*
4. *Let \mathcal{I}_i be the set of intervals in valuation space on which $G_i(F_i(\cdot)) > \bar{G}_i(F_i(\cdot))$.*

Definition 3.2.5 ($\bar{\mathcal{A}}$) *The ironed algorithm corresponding to algorithm \mathcal{A} is*

$$\bar{\mathcal{A}} = \text{RESAMPLE}(\mathcal{A}, \text{MONOINTS}(\mathbf{x})).$$

Lemma 3.2.6 $\bar{\mathcal{A}}$ *is monotone.*

Proof: We must show that each agent has a monotone allocation rule. The allocation rule for agent i is precisely $\bar{x}_i(v_i) = \bar{g}(F_i(v_i))$, which is the derivative of a convex function and therefore monotone. \square

Lemma 3.2.7 *If \mathcal{A} is a Bayesian β -approximation then $\bar{\mathcal{A}}$ is a Bayesian β -approximation.*

Proof: First notice that the two allocation rules produce the same distribution over allocations and therefore expected costs are identical. We will show, for a single agent i ,

that $\mathbf{E}[v_i \bar{x}_i(v_i)] \geq \mathbf{E}[v_i x_i(v_i)]$, from which linearity of expectation implies the result. We have

$$\begin{aligned} \mathbf{E}[v_i x_i(v_i)] &= \int_0^1 v x_i(v) f_i(v) dv = \int_0^1 F_i^{-1}(q) g_i(q) dq \\ &= \int_0^1 \int_0^{F_i^{-1}(q)} g_i(q) dz dq = \int_0^1 \int_{F_i(z)}^1 g_i(q) dq dz \\ &= \int_0^1 (G_i(1) - G_i(F_i(z))) dz \end{aligned}$$

and similarly $\mathbf{E}[v_i \bar{x}_i(v_i)] = \int_0^1 (\bar{G}_i(1) - \bar{G}_i(F_i(z))) dz$. We conclude $\mathbf{E}[v_i \bar{x}_i(v_i)] \geq \mathbf{E}[v_i x_i(v_i)]$ since $\bar{G}_i(1) = G_i(1)$ and $\bar{G}_i(F_i(z)) \leq G_i(F_i(z))$ for all $z \in [0, 1]$. \square

Theorem 3.2.1 follows from Lemmas 3.2.6 and 3.2.7.

3.2.3 Computational Issues

Let us make a few notes about the computational complexity of constructing the set of intervals $\text{MONOINTS}(\mathbf{x})$. Suppose, for instance, that the inputs are given as fixed-precision values with length polynomial in n (so, in particular, the input space is discretized), and we are given access to the value of $F_i(v_i)$ and $x_i(v_i)$ for all such values v_i . In this case, it is possible that $\text{MONOINTS}(\mathbf{x})$ consists of exponentially many intervals; for example, it might partition the input space into intervals that each contain two input values. However, it is not actually necessary for our transformation to compute the set of monotonicizing intervals $\text{MONOINTS}(\mathbf{x})$ ahead of time; it need only determine the interval in which each given input value resides. Thus, to implement $\bar{\mathcal{A}}$ efficiently, it is sufficient to be able to compute, for any v_i , the interval of $\text{MONOINTS}(\mathbf{x})$ containing v_i .

In general, this operation requires that we find a segment of the convex hull of an integral curve, which (in, say, a fixed-precision model) may have exponentially many points. However, we note that if we can assume that the input space for each agent is of polynomial size, this operation can be completed efficiently.

We will refrain from a formal treatment of the computational issues inherent in our

reduction, as they will be addressed in the more general black-box reduction described in Section 3.3.

3.2.4 Comparison with Myerson's Mechanism

At the heart of our mechanism construction is an ironing procedure that monotoneizes allocation rules. A similar process is used by Myerson as part of his construction of (revenue) optimal mechanisms for single-parameter settings [72]. Are these, in fact, the same mechanism? In light of their similarity, we will now compare these two constructions and highlight their differences.

We first recall Myerson's optimal mechanism. For each agent i , the mechanism considers the *virtual valuation function* $\phi_i(\cdot)$ given by $\phi_i(v_i) = v_i - \frac{1-F_i(v_i)}{f_i(v_i)}$. This function is monotone³ using the ironing method described in Section 3.2.1; the resulting monotone function is denoted $\bar{\phi}_i(\cdot)$. Given a valuation profile \mathbf{v} , the mechanism returns the allocation \mathbf{x} that maximizes $\sum_i \bar{\phi}_i(v_i) \cdot x_i - c(\mathbf{x})$. Myerson's celebrated result is that this allocation rule is revenue-optimal among the class of incentive compatible allocation rules [72].

Informally speaking, one can interpret Myerson's mechanism as considering the allocation rule that maximizes social welfare with respect to the profile of virtual values $\phi_i(v_i)$. If the virtual valuation function is non-monotone, this allocation rule will also be non-monotone and hence not incentive compatible. The mechanism addresses this issue by ironing the virtual valuation function, which effectively monotoneizes the allocation rule.

The motivation for ironing in our construction is quite similar, in that we are given a non-monotone allocation rule that we wish to make incentive compatible. Furthermore, we address the issue in a similar way: by ironing the offending non-monotone curve. One

³Note that the virtual valuation function may be non-monotone if F_i does not satisfy the monotone hazard rate assumption. For instance, bimodal distributions generally have non-monotone virtual valuation functions.

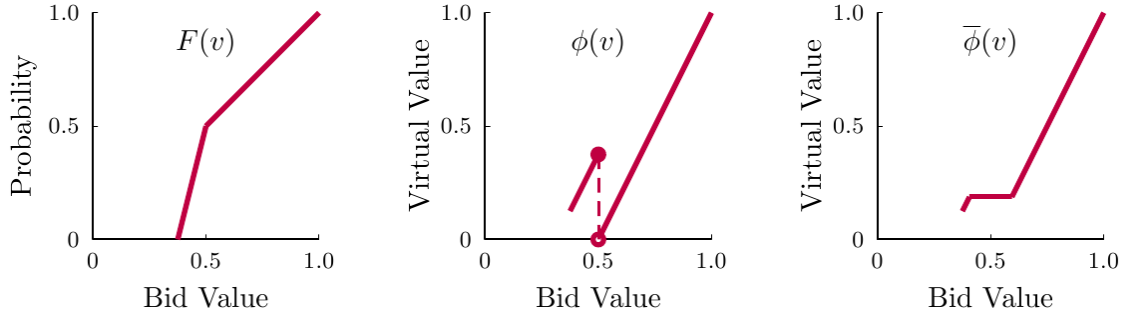


Figure 3.3: The distribution $F(v)$ used in Appendix 3.2.4, with virtual valuation function $\phi(v)$ and ironed virtual valuation function $\bar{\phi}(v)$.

might therefore suspect that these two monotization procedures are, in fact, equivalent when restricted to the allocation rule that maximizes virtual welfare. However, as we will now show, this is not the case: the mechanisms that result from ironing the virtual valuation function and from ironing the allocation rule are distinct. Thus, our construction does differ, in an essential way, from that of Myerson.

Let us provide an example to illustrate this difference. Consider an auction of a single indivisible item to multiple bidders with values drawn i.i.d. from distribution F . Consider the following distribution F : with probability $1/2$, the value is drawn uniformly from $[\frac{3}{8}, \frac{1}{2}]$; otherwise, it is drawn uniformly from $(\frac{1}{2}, 1]$ (see Figure 3.3). The virtual valuation function and ironed virtual valuation function corresponding to this distribution are

$$\phi(v) = \begin{cases} 2v - \frac{5}{8} & v \in [\frac{3}{8}, \frac{1}{2}] \\ 2v - 1 & v \in (\frac{1}{2}, 1] \end{cases} \quad \bar{\phi}(v) = \begin{cases} 2v - \frac{5}{8} & v \in [\frac{3}{8}, \frac{13}{32}] \\ \frac{3}{16} & v \in (\frac{13}{32}, \frac{19}{32}] \\ 2v - 1 & v \in (\frac{19}{32}, 1] \end{cases}$$

Suppose \mathcal{A} is the allocation rule that assigns the item to the agent with highest virtual value. We now consider the two incentive compatible variants of \mathcal{A} that we wish to compare. Namely, let \mathcal{A}' be Myerson's algorithm, which assigns the item to the agent with the highest *ironed* virtual value, and let $\bar{\mathcal{A}}$ be the ironed algorithm corresponding to

\mathcal{A} (as in Section 3.2). Let $x(\cdot)$, $x'(\cdot)$, and $\bar{x}(\cdot)$ denote the allocation curves corresponding to \mathcal{A} , \mathcal{A}' , and $\bar{\mathcal{A}}$, respectively⁴. Our goal is to show that $\bar{x}(\cdot) \neq x'(\cdot)$.

We observe that the function $\bar{\phi}(\cdot)$ achieves a strict minimum, over its effective range $[\frac{3}{8}, 1]$, at the point $v = \frac{3}{8}$. This implies that $x'(\frac{3}{8}) = 0$, since an agent that declares the minimal value can be awarded an allocation only in the 0-probability event that all other agents report this same value.

On the other hand, it must be that $\bar{x}(\frac{3}{8}) = \mathbf{E}_v[x(v) \mid v \leq z]$ for some $z \in [\frac{3}{8}, 1]$. We claim that this value is strictly positive. Indeed, $\phi(v) > \phi(w)$ for $v \in [\frac{3}{8}, \frac{1}{2}]$ and $w \in (\frac{1}{2}, \frac{9}{16})$. This implies that $x(v) > 0$ for all $v \in [\frac{3}{8}, \frac{1}{2}]$. We must therefore have $\bar{x}(\frac{3}{8}) = \mathbf{E}_v[x(v) \mid v \leq z] > 0$.

We conclude $x'(\frac{3}{8}) \neq \bar{x}(\frac{3}{8})$, and thus the allocation rules \mathcal{A}' and $\bar{\mathcal{A}}$ are distinct.

3.3 A Black-Box Reduction

The construction outlined in Section 3.2 succeeds in monotonicizing an approximation algorithm without loss, but suffers from two problems. First, it requires full functional access to the allocation rules and value distributions, which may be unreasonable in some settings (as it requires a full analysis of the behaviour of the algorithm in expectation over the prior distribution, for each agent). Second, it does not guarantee that the transformation π can be calculated in polynomial time, given an appropriate computational model. In this section we address both issues by implementing our construction as a polytime transformation that uses only black-box access to the algorithm \mathcal{A} and prior distribution \mathbf{F} . That is, our mechanism can sample from the prior distribution and sample the outcome of \mathcal{A} for any input vector (recalling that \mathcal{A} may be randomized), but otherwise has no information about the distribution, algorithm, or feasibility constraints of the problem.

⁴We drop the usual subscript of agent index since, by symmetry, the allocation curves are the same for each player.

We will use the ironing procedure from Section 3.2 to monotone an algorithm in this black-box model. Instead of using direct knowledge of the allocation rule, we must use sampling to estimate it. This sampling introduces errors in the selection of interval sets for resampling, which must then be dealt with. Our analysis will proceed in the following steps.

1. In Section 3.3.1, we describe a method for computing payments in the black-box model.
2. In Section 3.3.2, we describe a method for combining sampling with ironing to obtain a nearly monotone algorithm. In fact, this algorithm will be ϵ -Bayesian incentive compatible (to be defined below).
3. In Section 3.3.3, we show that a convex combination of this nearly monotone algorithm with a blatantly monotone one will give a monotone algorithm, resulting in a BIC mechanism.

All of these steps approximately preserve social welfare. We obtain the following theorem.

Theorem 3.3.1 *In the black-box model and for general cost settings, for any $\epsilon > 0$, a BIC algorithm \mathcal{A}' can be computed from any algorithm \mathcal{A} . Its expected social welfare satisfies $\mathcal{A}' \geq \mathcal{A} - \epsilon$, and its runtime is polynomial in n , $1/\epsilon$, and the runtime of \mathcal{A} .*

The additive error in Theorem 3.3.1 can be converted into a multiplicative error whenever the expected welfare of \mathcal{A} is not too small. We obtain the following corollary.

Corollary 3.3.2 *In the black-box model and general cost settings, for any $\epsilon > 0$, a BIC algorithm \mathcal{A}' can be computed from any algorithm \mathcal{A} . Its expected social welfare satisfies $\mathcal{A}' \geq \mathcal{A}/(1 + \epsilon)$, and its runtime is polynomial in n , $1/\epsilon$, $1/\mathcal{A}$, and the runtime of \mathcal{A} .*

Corollary 3.3.2 gives a construction with a multiplicative error in social welfare, but its runtime depends on the expected welfare of \mathcal{A} . In Section 3.3.4 we describe an improvement that removes this dependency, and implies a fully polynomial reduction for downward-closed settings.

3.3.1 Computing Payments

Suppose that \mathcal{A} has monotone allocation rules. The problem of designing a mechanism to implement \mathcal{A} then reduces to calculating appropriate payments. These payments are completely determined by the allocation rule of \mathcal{A} , but in the black-box model we do not have direct access to the functional form of the allocation rule. Archer et al. [3] solve this problem by computing an unbiased estimator of the desired payment rule using only black-box calls to the algorithm. For completeness we now summarize their approach.

Definition 3.3.3 (black-box payments) *If algorithm \mathcal{A} does not allocate to agent i , then agent i pays 0. Otherwise, we compute the payment of agent i as follows:*

1. Choose v_i' uniformly from $[0, v_i]$
2. Draw $\mathbf{v}'_{-i} \sim \mathbf{F}_{-i}$ and run $\mathcal{A}(v_i', \mathbf{v}'_{-i})$
3. If \mathcal{A} allocated to agent i in the previous step set $X = v_i$, otherwise set $X = 0$.
4. If $X \neq 0$, repeatedly draw values $\mathbf{v}'_{-i} \sim \mathbf{F}_{-i}$ and run $\mathcal{A}(v_i, \mathbf{v}'_{-i})$ until the algorithm allocates to player i , and let T be the number of iterations required.
5. Agent i 's payment is $p_i = v_i - TX$.

As was shown by Archer et al., this computation attains the appropriate expected payment.

Claim 3.3.4 (Archer et al. [3]) *In the black-box payment procedure, the expected payments are*

$$p_i(v_i) = v_i x_i(v_i) - \int_0^{v_i} x_i(z) dz.$$

We note that since we execute this procedure for agent i only if he receives an allocation, which occurs with probability $x_i(v_i)$, the expected number of calls to \mathcal{A} for each player is at most

$$x_i(v_i) \left(1 + \frac{1}{x_i(v_i)}\right) \leq 2.$$

Thus, in expectation, all payments are computed with $2n$ calls to \mathcal{A} .

Any mechanism paired with the above payment scheme will be *individually rational* (IR), meaning that a truthtelling agent will never obtain negative utility. This is true even if the allocation rule is not monotone. This follows immediately from the fact that the payment for an agent that declares value v_i is never greater than v_i (indeed, it is defined as v_i minus a non-negative value).

3.3.2 Sampling and Approximate Truthfulness

We will be estimating the allocation rule of a non-monotone algorithm and attempting to iron it. This will fail to result in an absolutely monotone rule. In this section we show that a nearly monotone rule results in truthtelling as an ϵ -Bayes-Nash equilibrium (ϵ -BNE): the most an agent can gain from a non-truthtelling strategy is an additive ϵ . We call such a mechanism ϵ -Bayesian incentive compatible (ϵ -BIC).

Definition 3.3.5 (ϵ -BIC) *A mechanism is ϵ -Bayesian incentive compatible if truthtelling obtains at least as much utility as any other strategy, up to an additive ϵ , assuming all other agents truthtell. That is, for all i , v_i , and v' , $v_i x_i(v_i) - p_i(v_i) \geq v_i x(v') - p_i(v') - \epsilon$.*

The main theorem of this section is the following.

Theorem 3.3.6 *In the black-box model and general cost settings, for any $\epsilon > 0$, an ϵ -BIC algorithm \mathcal{A}' can be computed from any algorithm \mathcal{A} . Its expected social welfare satisfies $\mathcal{A}' \geq \mathcal{A} - \epsilon$, and its runtime is polynomial in n and $1/\epsilon$.*

The resampling procedure from Section 3.2 is the main workhorse for Theorem 3.3.6. The construction of algorithm \mathcal{A}' consists primarily of choosing interval sets on which to resample.

ϵ -closeness

We now formalize a closeness property under which an allocation rule that is close to monotone is ϵ -BIC (for some related ϵ).

Definition 3.3.7 (ϵ -close) *Allocation rules $x(\cdot)$ and $x'(\cdot)$ are ϵ -close if $|x(v) - x'(v)| < \epsilon$ for all v . Two algorithms or mechanisms are ϵ -close if their corresponding allocation rules are ϵ -close for each agent.*

Lemma 3.3.8 *If non-monotone \mathcal{A}' is ϵ -close to a monotone \mathcal{A} , then \mathcal{A}' is (2ϵ) -BIC.*

Proof: Suppose agent i is participating in \mathcal{A}' and has value v_i , but claims to have value v_i' . Assume $v_i > v_i'$; the opposite case is similar. Using the payment rule from Theorem 2.5.1, agent i 's gain in utility from declaring v_i' is:

$$(v_i x_i'(v_i') - p_i(v_i')) - (v_i x_i'(v_i) - p_i(v_i)) = (v_i - v_i') x_i'(v_i') - \int_{v_i'}^{v_i} x_i'(z) dz. \quad (3.1)$$

Since $x_i'(\cdot)$ is ϵ -close to a monotone curve, it must be that $x_i'(z) + \epsilon \geq x_i'(v_i') - \epsilon$ for all $z \in [v_i', v_i]$. Thus $\int_{v_i'}^{v_i} x_i'(z) dz \geq (v_i - v_i')(x_i'(v_i') - 2\epsilon)$. This implies that the value in (3.1) is at most $2\epsilon(v_i - v_i')$, which is at most 2ϵ . \square

Lemma 3.3.9 *If \mathcal{A} and \mathcal{A}' have the same expected costs⁵ and are ϵ -close then $\mathcal{A}' \geq \mathcal{A} - n\epsilon$.*

Proof: For each agent i , $\mathbf{E}[v_i x_i'(v_i)] \geq \mathbf{E}[v_i(x_i(v_i) - \epsilon)] \geq \mathbf{E}[v_i x_i(v_i)] - \epsilon \mathbf{E}[v_i]$. The result then follows by linearity of expectation, plus the fact that $\mathbf{E}[v_i] \leq 1$. \square

⁵Recall that the *expected cost* of an algorithm A with allocation rule $\mathbf{x}(\cdot)$ is $\mathbf{E}_{\mathbf{v} \sim \mathbf{F}}[c(\mathbf{x}(\mathbf{v}))]$.

Lemma 3.3.10 *If algorithms \mathcal{A} and \mathcal{A}' are ϵ -close, then for any collection of interval sets $\mathcal{I} = \{\mathcal{I}_1, \dots, \mathcal{I}_n\}$, resampled algorithms $\text{RESAMPLE}(\mathcal{A}, \mathcal{I})$ and $\text{RESAMPLE}(\mathcal{A}', \mathcal{I})$ are ϵ -close.*

Proof: For any i , let x_i and x_i' be the allocation rules of \mathcal{A} and \mathcal{A}' , respectively. Let \bar{x}_i and \bar{x}_i' be the allocation rules of $\text{RESAMPLE}(\mathcal{A}, \mathcal{I})$ and $\text{RESAMPLE}(\mathcal{A}', \mathcal{I})$. Then for any $I \in \mathcal{I}_i$,

$$|\bar{x}_i(I) - \bar{x}_i'(I)| = |E_v[x(v) \mid v \in I] - E_v[x'(v) \mid v \in I]| = |E_v[x(v) - x'(v) \mid v \in I]| < \epsilon.$$

□

Appropriate payments to turn an algorithm that is ϵ -close to monotone into a mechanism that is 2ϵ -BIC can be computed by the same process we would use for monotone algorithms.

Discretization

A key step in our reduction will be in discretizing the allocation rules of the algorithm. This reduces the problem of estimating an allocation rule to estimating its value at a polynomial number of points. Moreover, our resulting allocation will not necessarily be monotone, but there will be only a polynomial number of points at which it can be non-monotone; we will use this to our advantage when fixing non-monotonicities in Section 3.3.3.

Definition 3.3.11 (Piecewise constant) *An algorithm is k -piece piecewise constant if for each i there is a partition of valuation space into at most k intervals such that the allocation rule for agent i is constant on each interval.*

Definition 3.3.12 ($\text{DISC}_\epsilon(\mathcal{A})$) *For a given $\epsilon > 0$ and algorithm \mathcal{A} , the discretization of algorithm \mathcal{A} , $\text{DISC}_\epsilon(\mathcal{A})$, is $\text{RESAMPLE}(\mathcal{A}, \mathcal{I})$, where $\mathcal{I} = \{\mathcal{I}_1, \dots, \mathcal{I}_n\}$ is the collection of intervals defined by*

$$\mathcal{I}_i = \{[0, \epsilon)\} \cup \{[\epsilon(1 + \epsilon)^t, \epsilon(1 + \epsilon)^{t+1})\}_{0 \leq t \leq \log_{1+\epsilon}(1/\epsilon)}.$$

Lemma 3.3.13 $\text{DISC}_\epsilon(\mathcal{A})$ is $\log_{1+\epsilon}(1/\epsilon)$ -piece piecewise constant and $\text{DISC}_\epsilon(\mathcal{A}) \geq \mathcal{A} - 2n\epsilon$.

Proof: Let $\dot{\mathbf{x}}(\cdot)$ denote the allocation rules for $\text{DISC}_\epsilon(\mathcal{A})$. The allocation curves for $\text{DISC}_\epsilon(\mathcal{A})$ are constant on interval $[0, \epsilon)$ and all intervals of the form $[\epsilon(1+\epsilon)^t, \epsilon(1+\epsilon)^{t+1})$, and there are at most $\log_{1+\epsilon}(\epsilon^{-1})$ such intervals over the range $[\epsilon, 1]$. These intervals do, indeed, partition valuation space. Furthermore, $\mathbf{E}_{v_i}[v_i \dot{x}_i(v_i)] \geq (1-\epsilon)\mathbf{E}_{v_i}[v_i x_i(v_i)] - \epsilon \geq \mathbf{E}_{v_i}[v_i x_i(v_i)] - 2\epsilon$, as algorithm $\text{DISC}_\epsilon(\mathcal{A})$ modifies any input value greater than ϵ by at most a factor of $(1-\epsilon)$. As the expected costs before and after discretization are the same, the result follows from linearity of expectation. \square

Statistical Estimation

We next describe a sampling procedure for estimating an allocation rule. This procedure will not form an algorithm, but rather generates an estimated allocation curve, which we will denote by $\mathbf{y}(\cdot)$. This estimate behaves like an allocation rule, but is not associated with an actual algorithm (and, in particular, need not be feasibly implementable).

Definition 3.3.14 (estimate allocation rule) *Given an algorithm \mathcal{A} which is k -piece piecewise constant and $\epsilon > 0$, an estimated allocation rule for \mathcal{A} is a curve $\mathbf{y}(\cdot)$ found as follows:*

1. for each agent i and valuation-space piece I_j , draw $\frac{4}{\epsilon^2} \log(2kn/\epsilon)$ samples from \mathbf{F} conditional on $v_i \in I_j$, and run \mathcal{A} on each of these samples.
2. let y_{ij} be the average allocation over the invocations to \mathcal{A} above, for each i and j .
3. Define \mathbf{y} by $y_i(v) = y_{ij}$ for all $v \in I_j$

Lemma 3.3.15 *If algorithm \mathcal{A} is k -piece piecewise constant then, for any $\epsilon > 0$, an estimated allocation rule $\mathbf{y}(\cdot)$ for \mathcal{A} is k -piece piecewise constant, and is $\frac{\epsilon}{2}$ -close to $\mathbf{x}(\cdot)$*

with probability at least $1 - \frac{\epsilon}{2}$. The number of black-box calls to \mathcal{A} used in the construction of $\mathbf{y}(\cdot)$ is polynomial in n , k , and $1/\epsilon$.

Proof: The runtime bound and the fact that $\mathbf{y}(\cdot)$ is k -piece piecewise constant follow immediately from the definition of $\mathbf{y}(\cdot)$. Choose some i and let I_j denote piece j of the valuation space for agent i in \mathcal{A} , and write $x_i(I_j)$ for the (constant) value of $x_i(v)$ for any $v \in I_j$. By the Hoeffding-Chernoff inequality,

$$\Pr[|y_{ij} - x_i(I_j)| > \epsilon/2] \leq e^{-4(\epsilon)^{-2} \log(2kn/\epsilon)(\epsilon/2)^2} \leq \epsilon/2kn.$$

Thus, taking the union bound over all i and j , we conclude that

$$|y_{ij} - x_i(I_j)| \leq \frac{\epsilon}{2}$$

for all i and j with probability at least $1 - \frac{\epsilon}{2}$. \square

We now complete the proof of Theorem 3.3.6 by combining our sampling procedure with the ironing procedure from the ideal model.

Definition 3.3.16 ($\text{IRON}_\epsilon(\mathcal{A})$) *Given piecewise constant algorithm \mathcal{A} , the statistically ironed algorithm for \mathcal{A} with error $\epsilon > 0$ is $\text{IRON}_\epsilon(\mathcal{A}) = \text{RESAMPLE}(\mathcal{A}, \text{MONOINTS}(\mathbf{y}))$ where $\mathbf{y}(\cdot)$ is the estimated allocation rule for \mathcal{A} and ϵ .*

Note that $\text{IRON}_\epsilon(\mathcal{A})$ is not simply a resampling of \mathcal{A} , but rather a convex combination of resamplings since the construction of interval set $\text{MONOINTS}(\mathbf{y})$ is randomized.

Lemma 3.3.17 $\text{IRON}_\epsilon(\mathcal{A})$ *is 2ϵ -BIC and $\text{IRON}_\epsilon(\mathcal{A}) \geq \mathcal{A} - n\epsilon$.*

Proof: By Lemma 3.3.15, $y_i(\cdot)$ is k -piece piecewise constant for each i . Let $\mathcal{A}_{\mathbf{y}}$ be the (fictional) algorithm with allocation rule \mathbf{y} . Since \mathcal{I} is the monotonicizing interval set for $\mathcal{A}_{\mathbf{y}}$, if $\mathcal{A}_{\mathbf{y}}$ were ironed according to \mathcal{I} , the result would be $\bar{\mathcal{A}}_{\mathbf{y}}$ which is monotone.

By Lemma 3.3.15, $\mathcal{A}_{\mathbf{y}}$ is $\frac{\epsilon}{2}$ -close to \mathcal{A} with probability $1 - \frac{\epsilon}{2}$. In this case, Lemma 3.3.10 implies $\text{RESAMPLE}(\mathcal{A}, \mathcal{I})$ is $\frac{\epsilon}{2}$ -close to $\bar{\mathcal{A}}_{\mathbf{y}}$. For the remaining probability, $\frac{\epsilon}{2}$, we

note that $\text{RESAMPLE}(\mathcal{A}, \mathcal{I})$ is trivially 1-close to $\bar{\mathcal{A}}_{\mathbf{y}}$. Thus, taking expectation over all possible outcomes of the sampling, we conclude that $\text{IRON}_{\epsilon}(\mathcal{A})$ is ϵ -close to monotone, and is therefore 2ϵ -BIC by Lemma 3.3.8.

Since, with probability $1 - \frac{\epsilon}{2}$, $\text{RESAMPLE}(\mathcal{A}, \mathcal{I})$ is $\frac{\epsilon}{2}$ close to $\bar{\mathcal{A}}_{\mathbf{y}}$ and $\mathcal{A}_{\mathbf{y}}$ is $\frac{\epsilon}{2}$ close to \mathcal{A} , Lemma 3.3.9 and Lemma 3.3.13 imply that, with probability $1 - \frac{\epsilon}{2}$,

$$\text{RESAMPLE}(\mathcal{A}, \mathcal{I}) \geq \bar{\mathcal{A}}_{\mathbf{y}} - \frac{1}{2}n\epsilon \geq \mathcal{A}_{\mathbf{y}} - \frac{1}{2}n\epsilon \geq \mathcal{A} - n\epsilon.$$

For the remaining probability, $\frac{\epsilon}{2}$, we note that trivially $\text{RESAMPLE}(\mathcal{A}, \mathcal{I}) \geq 0 = \mathcal{A} - \mathcal{A} \geq \mathcal{A} - n$. Thus, taking expectation over all possible outcomes of sampling, we conclude $\text{IRON}_{\epsilon}(\mathcal{A}) \geq \mathcal{A} - n\epsilon$. \square

We are now ready to complete the proof of Theorem 3.3.6.

Proof of Theorem 3.3.6: Define \mathcal{A}' to be the algorithm $\text{IRON}_{\epsilon'}(\text{DISC}_{\epsilon'}(\mathcal{A}))$, where $\epsilon' = \epsilon/3n$. Then, by Lemmas 3.3.13 and 3.3.17, \mathcal{A}' is $2\epsilon'$ -BIC, and hence ϵ -BIC, and $\mathcal{A}' \geq \text{DISC}_{\epsilon'}(\mathcal{A}) - n\epsilon' \geq \mathcal{A} - 3n\epsilon' = \mathcal{A} - \epsilon$. The runtime of \mathcal{A}' (which is dominated by sampling in the construction of \mathbf{y}) is $O(nk\epsilon'^{-2} \log(2kn/\epsilon')) = \tilde{O}(n^3\epsilon^{-3} \log(\epsilon^{-1}))$, where recall $k = \frac{1}{\epsilon} \log(1/\epsilon)$ is the number of discrete intervals in $\text{DISC}_{\epsilon'}(\mathcal{A})$. \square

3.3.3 Exact Truthfulness

In the previous section we showed how to construct an ϵ -BIC mechanism from any algorithm with almost no loss to the social welfare. Our goal now is to take such an ϵ -BIC algorithm \mathcal{A} and make it BIC. In other words, we would like to “fix” the (small) non-monotonicities in \mathcal{A} . Fortunately, since each allocation curve of \mathcal{A} is discretized, any non-monotonicities must occur only at a small number of predetermined points. Our approach for removing these points of non-monotonicity is simple: we will construct an alternative algorithm \mathcal{A}' whose allocation curves are stair functions, with jumps in allocation probability occurring at each of those points. A convex combination of \mathcal{A} and \mathcal{A}' will then be monotone. This convex combination will be our final BIC algorithm.

It is important that this convex combination process not reduce social welfare by too much. This requires two things. First, we need the convex combination to be mostly \mathcal{A} as only it has provably good welfare. This is possible by taking ϵ so small that the explicit monotonicities in \mathcal{A}' heavily outweigh the non-monotonicities in \mathcal{A} (which are at most ϵ). Second, we need to ensure that the expected social welfare of \mathcal{A}' is not extremely negative.

How should we construct \mathcal{A}' ? Suppose first that we are in a downward-closed feasibility setting. In this case, the singleton allocation $\{i\}$ is feasible for each (non-trivial) agent i . The construction of \mathcal{A}' with stair-function allocation curves is then straightforward: an agent i is chosen uniformly at random and the algorithm then either allocates to agent i or not, with the probability of allocation following a stair function. Since \mathcal{A}' only returns feasible outcomes, its expected social welfare must be non-negative.

We would like to follow this same approach in general cost settings. However, it may be that, for some i , the particular allocation $\{i\}$ has an extremely high (or infinite) cost, in which case the above algorithm may have an extremely negative social welfare. Note, though, that in our construction we can replace $\{i\}$ with *any* allocation that includes agent i . It is therefore sufficient to find, for each i , some allocation that includes agent i and whose cost is not too high. Once these allocations are found, we can use them to construct the stair algorithm \mathcal{A}' .

In some cases finding low-cost allocations may be highly non-trivial. To get around this problem, we observe that as long as algorithm \mathcal{A} has a reasonable probability of allocating to agent i , there must exist low-cost allocations that include i that are returned by \mathcal{A} . We can therefore find such allocations by repeatedly sampling outcomes of \mathcal{A} . If, on the other hand, we were to take many samples and not find any allocations that include agent i , then we can safely assume that agent i does not contribute much to the expected social welfare of \mathcal{A} . In this case, we can trivially monotinize agent i 's allocation curve by ironing on interval $[0, 1]$, removing the need to find allocations that

include agent i .

The Stair Algorithm

We begin by demonstrating how to combine an ϵ -BIC mechanism with an algorithm whose allocation rules are stair functions in order to obtain a BIC mechanism.

Definition 3.3.18 ($\text{STAIR}(\mathcal{A})$) *Let \mathcal{A} be a k -piece piecewise constant algorithm, and suppose S_1, \dots, S_n and T_1, \dots, T_n are allocations such that $i \in S_i$ and $i \notin T_i$ for all i . The stair algorithm for \mathcal{A} , $\text{STAIR}(\mathcal{A})$, does the following:*

1. *Pick an agent i uniformly from the n agents.*
2. *If v_i is in the j th highest piece of k pieces, allocate to S_i with probability $(j-1)/(k-1)$ and T_i otherwise.*

Definition 3.3.19 ($\text{COMB}_\epsilon(\mathcal{A})$) *Suppose algorithm \mathcal{A} is k -piece piecewise constant. We then write $\text{COMB}_\epsilon(\mathcal{A})$ for the convex combination of \mathcal{A} with probability $1-\delta$ and $\text{STAIR}(\mathcal{A})$ with probability δ , where $\delta = 2(k-1)n\epsilon$.*

Lemma 3.3.20 *If \mathcal{A} is ϵ -close to a monotone \mathcal{A}' , then algorithm $\text{COMB}_\epsilon(\mathcal{A})$ is BIC.*

Proof: We will write $\hat{x}_i(\cdot)$ to denote an allocation rule of $\text{COMB}_\epsilon(\mathcal{A})$. To show $\text{COMB}_\epsilon(\mathcal{A})$ is BIC, choose any agent i and any values $v_i < v_i'$; we will show $\hat{x}_i(v_i) \leq \hat{x}_i(v_i')$. If v_i, v_i' are in the same piece of the valuation space then $\hat{x}_i(v_i) = \hat{x}_i(v_i')$. Otherwise, since \mathcal{A} is ϵ -close to monotone \mathcal{A}' , it must be that $x_i(v_i) \leq x_i(v_i') - 2\epsilon$. Furthermore, if $\mathbf{s}(\cdot)$ is the allocation rule for $\text{STAIR}(\mathcal{A}')$, then $s_i(v_i) \leq s_i(v_i') + 1/(k-1)n$. We conclude that

$$\begin{aligned} \hat{x}_i(v_i) &= (1-\delta)x_i'(v_i) + \delta s_i(v_i) \\ &\leq \hat{x}_i(v_i') - 2\epsilon + \delta/(k-1)n \\ &= \hat{x}_i(v_i') \end{aligned}$$

as required, since $\delta = 2(k-1)n\epsilon$. □

Bounding Social Welfare: Finding Low-Cost Sets

We now describe the choice of sets S_1, \dots, S_n and T_1, \dots, T_n for algorithm STAIR(\mathcal{A}). What we require is that, for all i , we have $i \in S_i$, $i \notin T_i$, and S_i, T_i are feasible (or have sufficiently low cost). In many settings finding such sets is trivial (e.g., for downward-closed feasibility problems we can take $S_i = \{i\}$ and $T_i = \emptyset$), but for some problems it might be difficult to find feasible (or low-cost) allocations. Our approach is as follows. Since \mathcal{A} never makes an allocation that generates negative social welfare, we can bound the cost of any allocation made by \mathcal{A} . This motivates us to look for a set $S_i \ni i$ returned by \mathcal{A} on some input, for each i . This can be accomplished by sampling. This operation can be viewed as trimming away agents that are very rarely allocated. The same holds for finding T_i .

Definition 3.3.21 ($\text{TRIM}_\epsilon(\mathcal{A})$) *The trimmed algorithm for piece-wise constant \mathcal{A} is $\text{TRIM}_\epsilon(\mathcal{A})$:*

1. *For each agent i and valuation-space piece $I_j \in \mathcal{I}_i$, draw $\frac{4}{\epsilon^2} \log(4n/\epsilon)$ samples from \mathbf{F} conditional on $v_i \in I_j$, and run \mathcal{A} on each of these samples.*
2. *If \mathcal{A} is the same (always or never allocating) for i on every sample, define $\mathcal{I}'_i = \{[0, 1]\}$; otherwise, $\mathcal{I}'_i = \mathcal{I}_i$ and we define S_i to be any observed allocation that includes agent i and T_i to be any observed allocation that does not include agent i .*
3. *Run $\text{RESAMPLE}(\mathcal{A}, \mathcal{I}')$.*

Note that, for each i , either sets $S_i \ni i$ and $T_i \not\ni i$ will be found during the execution of $\text{TRIM}_\epsilon(\mathcal{A})$, or else the allocation rule of agent i will be made constant.

Lemma 3.3.22 $\text{TRIM}_\epsilon(\mathcal{A}) \geq \mathcal{A} - n\epsilon$.

Proof: We claim that, with probability at least $1 - \frac{\epsilon}{2}$, then the allocation rules for $\text{TRIM}_\epsilon(\mathcal{A})$ and \mathcal{A} will be identical for each agent i such that $x_i(v_i) > \frac{\epsilon}{2}$ for some v_i and

$x_i(v_i') < 1 - \frac{\epsilon}{2}$ for some v_i' . Before proving the claim, let us see how it implies the desired result. With probability $1 - \frac{\epsilon}{2}$, $\text{TRIM}_\epsilon(\mathcal{A})$ and \mathcal{A} will differ for agent i only if $x_i(v_i) \leq \frac{\epsilon}{2}$ for all v_i or $x_i(v_i) \geq 1 - \frac{\epsilon}{2}$ for all v_i . In this case, $\max_v \{x_i(v_i)\} - \min_v \{x_i(v_i)\} \leq \frac{\epsilon}{2}$. This implies that \mathcal{A} is $\frac{\epsilon}{2}$ -close to $\text{RESAMPLE}(\mathcal{A}, \mathcal{I}')$ for agent i , and thus $\text{TRIM}_\epsilon(\mathcal{A}) \geq \mathcal{A} - (\frac{\epsilon}{2})n$ with probability $1 - \frac{\epsilon}{2}$. For the remaining probability, we note that $\text{TRIM}_\epsilon(\mathcal{A}) \geq 0 = \mathcal{A} - \mathcal{A} \geq \mathcal{A} - n$ trivially. Thus, over all possible outcomes of sampling, we conclude that

$$\text{TRIM}_\epsilon(\mathcal{A}) \geq \mathcal{A} - \frac{\epsilon}{2}n - \frac{\epsilon}{2}n = \mathcal{A} - n\epsilon$$

as required.

Let us now prove the claim. Choose some agent i and suppose that $\text{TRIM}_\epsilon(\mathcal{A})$ and \mathcal{A} differ for agent i , and that there exist intervals $I_j, I_k \in \mathcal{I}_i$ with $x_i(I_j) \geq \frac{\epsilon}{2}$ and $x_i(I_k) \leq 1 - \frac{\epsilon}{2}$. Then, by the definition of \mathcal{I}'_i , it must be that no set $S \ni i$ was found during the sampling of interval I_j and no set $T \not\ni i$ was found during the sampling of interval I_k for agent i . However, since $x_i(I_j) \geq \frac{\epsilon}{2}$, there is a probability of at least $\frac{\epsilon}{2}$ of finding a set $S \ni i$ on each sample. By Chernoff-Hoeffding inequality, the probability that we do not find even one such set during $4\epsilon^{-2} \log(4n/\epsilon)$ samples is at most $\frac{\epsilon}{4n}$. Similarly, the probability of not finding a set $T \not\ni i$ on any sample of I_k is at most $\frac{\epsilon}{4n}$. We conclude that the probability that no set $S \ni i$ or $T \not\ni i$ was found during the sampling for agent i is at most $\frac{\epsilon}{2n}$. This is therefore a bound on the probability that $\text{TRIM}_\epsilon(\mathcal{A})$ and \mathcal{A} differ for agent i . By the union bound, the probability that this occurs for *any* agent is at most $\frac{\epsilon}{2}$, as required. \square

We are now ready to combine our tools into a BIC mechanism, proving Theorem 3.3.1.

Definition 3.3.23 ($\text{MONO}_\epsilon(\mathcal{A})$) *Given an algorithm \mathcal{A} and $\epsilon > 0$, the monotonization of \mathcal{A} , denoted $\text{MONO}_\epsilon(\mathcal{A})$, is the algorithm $\text{COMB}_\epsilon(\text{IRON}_\epsilon(\text{TRIM}_\epsilon(\text{DISC}_\epsilon(\mathcal{A}))))$.*

Lemma 3.3.24 $\text{MONO}_\epsilon(\mathcal{A})$ *is BIC, and $\text{MONO}_\epsilon(\mathcal{A}) \geq \mathcal{A} - 6kn^2\epsilon$.*

Proof: For notational convenience we define $\mathcal{A}' = \text{TRIM}_\epsilon(\text{DISC}_\epsilon(\mathcal{A}))$. Recall that during the construction of \mathcal{A}' we find sets S_1, \dots, S_n and T_1, \dots, T_n with $S_i \ni i$ and $T_i \not\ni i$. Also, Lemma 3.3.17 implies that $\text{IRON}_\epsilon(\mathcal{A}')$ is ϵ -close to a monotone algorithm. Thus $\text{COMB}_\epsilon(\text{IRON}_\epsilon(\mathcal{A}'))$ is well-defined, and is also BIC by Lemma 3.3.20.

Our ironing techniques do not affect the distribution of allocations generated by an algorithm, so the expected costs of $\text{MONO}_\epsilon(\mathcal{A})$ and \mathcal{A} are the same. Furthermore, by Lemmas 3.3.13, 3.3.17, and 3.3.22,

$$\text{IRON}_\epsilon(\mathcal{A}') \geq \mathcal{A}' - n\epsilon = \text{TRIM}_\epsilon(\text{DISC}_\epsilon(\mathcal{A})) - n\epsilon \geq \text{DISC}_\epsilon(\mathcal{A}) - 2n\epsilon \geq \mathcal{A} - 4n\epsilon.$$

We next claim that one can assume without loss of generality that $c(S_i) \leq n$ for all i . This is because S_i is in the range of \mathcal{A} , and we can assume that \mathcal{A} never returns an allocation that results in negative welfare (since otherwise a trivial improvement to \mathcal{A} would return the empty allocation instead). Since valuations lie in $[0, 1]$, non-negative welfare can be generated only by sets with cost at most n , and thus we can assume $c(S_i) \leq n$ for all i .

This implies that the expected social welfare obtained by $\text{STAIR}(\text{IRON}_\epsilon(\mathcal{A}'))$ is at least $(-n)$. We conclude

$$\begin{aligned} \text{MONO}_\epsilon(\mathcal{A}) &= \text{COMB}_\epsilon(\text{IRON}_\epsilon(\mathcal{A}')) \\ &= (1 - \delta)\text{IRON}_\epsilon(\mathcal{A}') - \delta\text{STAIR}(\text{IRON}_\epsilon(\mathcal{A}')) \\ &\geq \mathcal{A} - 4n\epsilon - (2(k - 1)n\epsilon)n \\ &\geq \mathcal{A} - 6kn^2\epsilon. \end{aligned}$$

□

Proof of Theorem 3.3.1: Let \mathcal{A}' be the monotonized algorithm $\text{MONO}_{\epsilon'}(\mathcal{A})$, where $\epsilon' = \epsilon/6kn^2$. The result then follows immediately from Lemma 3.3.24. The runtime, which is dominated by sampling, is $O(kn(\epsilon')^{-2}) = \tilde{O}(\frac{n^5}{\epsilon^5} \log^3(1/\epsilon))$. □

3.3.4 Multiplicative Error

We now show how to modify our construction from Section 3.3.3 to obtain a multiplicative error for many problems of interest, such as downward-closed feasibility settings. To see how this contrasts with Theorem 3.3.1, consider a setting in which the expected valuation of each agent, $\mathbf{E}[v_i]$, is exponentially small⁶. In this case, any additive error ϵ that is only polynomially small will dominate the expected welfare of algorithm \mathcal{A} . We therefore require a more general theorem in order to obtain meaningful results in this setting.

One might hope to obtain a reduction such that any algorithm \mathcal{A} can be converted into a BIC algorithm $\bar{\mathcal{A}}$ such that $\bar{\mathcal{A}} \geq (1 - \epsilon)\mathcal{A}$. However, we note that such a result would present difficulties for our sampling-based approach. For example, suppose that, on every input, algorithm \mathcal{A} returns a non-empty allocation with only an exponentially small probability. In this case, our mechanism cannot hope to learn anything useful about the algorithm given only a polynomial number of samples. Our additive-error reduction would simply iron each agents' entire value space aggressively, since doing so reduces the expected social welfare by only an exponentially small amount. However, in order to guarantee only a multiplicative loss in social welfare, the mechanism cannot iron so aggressively as to disrupt the (exponentially small) chance of a valuable allocation.

In light of this difficulty, we will limit our attention to algorithms that allocate to agents with no less than a polynomially small probability. To this end, let $\mu_{\max} = \max_i \mathbf{E}[v_i]$ be the maximum expected valuation of any agent. The following is a tightened version of Theorem 3.3.1 in which the loss in social welfare is scaled by μ_{\max} .

Theorem 3.3.25 *In the black-box model and general cost settings, for any $\epsilon > 0$, a BIC algorithm \mathcal{A}' can be computed from any Bayesian algorithm \mathcal{A} . Its social welfare satisfies $\mathcal{A}' \geq \mathcal{A} - \epsilon\mu_{\max}$, and its runtime is polynomial in n , $1/\epsilon$, and $\log(1/\mu_{\max})$.*

Thus, for any algorithm whose expected social welfare is polynomial in μ_{\max} (that is,

⁶Since values are scaled to lie in $[0, 1]$, this situation can occur whenever the expected valuations of the agents are bounded, but agents can have exponentially larger values with positive probability.

not super-polynomially smaller than μ_{\max}), we obtain a reduction with a multiplicative error in social welfare.⁷ This will be true of any algorithm that allocates to each agent with at least a polynomially small probability. For the special case of downward-closed set systems for feasibility problems, we can assume that $\mathcal{A} \geq \mu_{\max}$, since the trivial algorithm that simply allocates to the single player with the highest input value attains this value. This implies the following corollary.

Corollary 3.3.26 *In the black-box model and downward-closed feasibility settings, for any $\epsilon > 0$, a BIC Bayesian $\beta(1 + \epsilon)$ -approximation algorithm \mathcal{A}' can be computed from any Bayesian β -approximation algorithm \mathcal{A} . Its runtime is polynomial in n , $1/\epsilon$, and $\log(1/\mu_{\max})$.*

To prove Theorem 3.3.25, we first consider the following variant of Theorem 3.3.6:

Theorem 3.3.27 *In the black-box model and general cost settings, for any $\epsilon > 0$, an ϵ -BIC algorithm \mathcal{A}' can be computed from any Bayesian algorithm \mathcal{A} . Its social welfare satisfies $\mathcal{A}' \geq \mathcal{A} - \epsilon\mu_{\max}$, and its running time is polynomial in n , $1/\epsilon$, and $\log(1/\mu_{\max})$.*

The proof of Theorem 3.3.27 follows the proof of Theorem 3.3.6 from Section 3.3.2 almost exactly. Indeed, the only changes required are to replace instances of the inequality $\mathbf{E}[v_i] \leq 1$ with $\mathbf{E}[v_i] \leq \mu_{\max}$ throughout, and to alter the definition of the discretization of an algorithm \mathcal{A} , Definition 3.3.12, so that discretization occurs on the intervals

$$\mathcal{I}_i = \{[0, \epsilon\mu_{\max})\} \cup \{[\epsilon\mu_{\max}(1 + \epsilon)^t, \epsilon\mu_{\max}(1 + \epsilon)^{t+1})\}_{0 \leq t \leq \log_{1+\epsilon}(1/\epsilon\mu_{\max})}.$$

Lemmas 3.3.9, 3.3.13, and 3.3.15 then follow as before, with additive errors scaled by μ_{\max} . We omit further details of the proof of Theorem 3.3.27.

We now turn to proving Theorem 3.3.25 from Theorem 3.3.27. Our approach will be the same as the proof of Theorem 3.3.1 from Theorem 3.3.6 in Section 3.3.3: we consider

⁷Note that the result would also hold if we express the error as a multiple of the *average* expected value, rather than the maximum expected value, as these quantities differ by at most a polynomial factor.

a convex combination of the almost-monotone algorithm from Theorem 3.3.27 with the blatantly monotone stair algorithm. Recall that in Section 3.3.3 some care was necessary when finding sets S_1, \dots, S_n and T_1, \dots, T_n . This task becomes much more difficult when we wish to keep our error bounded by $\epsilon\mu_{\max}$ (rather than ϵ). We describe our approach in the next two subsections: we first give an algorithm for general cost settings, then present an optimization for the special case of feasibility settings.

Finding Low-Cost Sets

In general cost settings, algorithm $\text{STAIR}(\mathcal{A})$ may incur negative value if, for some i , the cost of set S_i or T_i is very large. Previously, it was sufficient to use the trivial bound that no allocation returned by \mathcal{A} can have cost larger than n , so that $\text{STAIR}(\mathcal{A})$ must generate welfare at least $-n$ (i.e. incur a loss of at most n). In order to obtain the improved result of Theorem 3.3.25, we must improve this bound.

Our goal will be for algorithm $\text{STAIR}(\mathcal{A})$ to have a welfare of at least $-n\mu_{\max}/\sqrt{\epsilon}$. Our approach will be to argue that an allocation with cost larger than $n\mu_{\max}/\sqrt{\epsilon}$ can be returned by \mathcal{A} only when the total value of all agents allocated to is greater than $n\mu_{\max}/\sqrt{\epsilon}$. However, if the expected value of each agent is at most μ_{\max} , then this occurs with small probability.

Observation 3.3.28 *For any agent i , $\Pr_{\mathbf{v}_{-i}}[\sum_{j \neq i} v_j \geq n\mu_{\max}/\sqrt{\epsilon}] < \sqrt{\epsilon}$.*

Proof: This follows directly from Markov's inequality, since the expected value of $\sum_{j \neq i} v_j$ is no greater than $n\mu_{\max}$. \square

Thus, if we take sufficiently many samples, we should expect to find examples of sets S_i and T_i , and moreover the costs of these sets should not be very large relative to $n\mu_{\max}$. There is, however, an additional complication. Recall that when we search for a set S_i (in $\text{TRIM}_\epsilon(\mathcal{A})$) we sample each interval in the domain of agent i . It may be that we find candidates for set S_i , but only in intervals with values much larger than μ_{\max} . In this

case, since $i \in S_i$, it may be that the cost of S_i is very large, which is allowed since agent i is assumed to have a large value. This presents a problem if we attempt to use S_i in algorithm $\text{STAIR}(\mathcal{A})$, since we allocate S_i with positive probability for all v_i that do not lie in the lowest-valued interval.

To get around the above problem, we apply the following trick. When choosing set S_i , we will favor sets found in lower-valued intervals. Thus, if set S_i is chosen from interval I_j , then it must be that no set $S \ni i$ was found in any interval I_k with $k < j$. With high probability this will only occur if $x_i(I_k)$ is very small for all $k < j$. We can therefore *iron together all intervals to the left of I_j* with minimal loss to the social welfare of the algorithm. Once this is done, I_j will be the second-smallest-valued interval for agent i , and therefore $\text{STAIR}(\mathcal{A})$ will allocate set S_i with positive probability only when agent i declares value from interval I_j or higher. Thus, even if the cost of S_i is very large due to the large values in interval I_j , this will be offset by the value obtained by agent i whenever set S_i is allocated by $\text{STAIR}(\mathcal{A})$.

We now proceed with implementing the above intuition more formally. Our upper bound on cost will depend on the following quantity, which relates to the structure of the piecewise constant intervals for algorithm \mathcal{A} .

Definition 3.3.29 *Suppose \mathcal{A} has piece-wise constant allocation rules, where the constant intervals for agent i are $\mathcal{I}_i = \{I_1, I_2, \dots\}$. The stair threshold for agent i , $w_i^{\mathcal{A}}$, is defined as $w_i^{\mathcal{A}} := \max I_1$ if $|\mathcal{I}_i| > 1$; otherwise $w_i^{\mathcal{A}} := \infty$. That is, $w_i^{\mathcal{A}}$ is the upper endpoint of the first valuation space interval for agent i , assuming the presence of multiple intervals.*

We can now relate the value of $\text{COMB}_\epsilon(\mathcal{A})$ to the cost of sets S_1, \dots, S_n and the stair thresholds of algorithm \mathcal{A} .

Lemma 3.3.30 *If there exists $X \geq 0$ such that $c(T_i) \leq X$ and $c(S_i) \leq w_i^{\mathcal{A}} + X$ for all i , then $\text{COMB}_\epsilon(\mathcal{A}) \geq \mathcal{A} - \delta(n\mu_{\max} + X)$.*

Proof: By construction, $\text{COMB}_\epsilon(\mathcal{A}) = (1 - \delta)\mathcal{A} + \delta\text{STAIR}(\mathcal{A})$. Recall that $\text{STAIR}(\mathcal{A})$ chooses some i uniformly at random, and then either allocates S_i or T_i . Moreover, $\text{STAIR}(\mathcal{A})$ will always allocate T_i if v_i is in the first piece of the valuation space; that is, if $v_i < w_i^{\mathcal{A}}$. We therefore conclude that $\text{STAIR}(\mathcal{A}) \geq \frac{1}{n} \sum_i \min\{w_i^{\mathcal{A}} - c(S_i), -c(T_i)\} \geq -X$. Also, $\mathcal{A} \leq n\mu_{\max}$ trivially. Thus $\text{COMB}_\epsilon(\mathcal{A}) \geq \mathcal{A} - \delta n\mu_{\max} + \delta(-X) = \mathcal{A} - \delta(n\mu_{\max} + X)$. \square

Our goal will be to find sets T_i with $c(T_i) \leq n\mu_{\max}/\sqrt{\epsilon}$ and S_i with $c(S_i) \leq w_i^{\mathcal{A}} + n\mu_{\max}/\sqrt{\epsilon}$, then apply Lemma 3.3.30 with $X = n\mu_{\max}/\sqrt{\epsilon}$. To find such sets, we will apply the same sampling techniques used in the construction of $\text{IRON}_\epsilon(\mathcal{A})$. That is, for each i and each piece of the valuation space, we will run \mathcal{A} on many sample inputs. As long as $x_i(I)$ is not too small on a given interval I , we are very likely to find some valid allocation that includes agent i during the sampling process! Moreover, we will show that not all sets discovered in this way can have high cost, so with high probability we will find sets T_i and S_i with cost at most $n\mu_{\max}/\sqrt{\epsilon}$. To relate the cost of set S_i with $w_i^{\mathcal{A}}$, we will also *iron together all left-most intervals for which a low-cost set was not found*. These ironed-together intervals will then act like a single piece of valuation space, which will allow us to relate the cost of any set S_i we *do* find to the stair threshold for the (modified) algorithm.

Definition 3.3.31 ($\text{TRIM}_\epsilon(\mathcal{A})$) *Given piece-wise constant algorithm \mathcal{A} , the stair compatible algorithm for \mathcal{A} , $\text{TRIM}_\epsilon(\mathcal{A})$, is as follows:*

1. *For each agent i :*
2. *Let $\mathcal{I}_i = \{I_1, \dots, I_k\}$ be the constant valuation space intervals for agent i .*
3. *For each $I_j \in \mathcal{I}_i$, draw $4\epsilon^{-2} \log(4n/\epsilon)$ samples from \mathbf{F} conditional on $v_i \in I_j$, and run \mathcal{A} on each of these samples.*
4. *Choose T_i to be any set returned by \mathcal{A} on some sample with $T_i \not\ni i$ and $c(T_i) \leq n\mu_{\max}/\sqrt{\epsilon}$.*

5. Let j_i be the minimal index such that, for some sample of interval I_{j_i} , \mathcal{A} allocated a set S with $S \ni i$ and $c(S) \leq \min I_{j_i} + n\mu_{\max}/\sqrt{\epsilon}$. Choose S_i to be any such S .
6. If no such sets S_i or T_i were returned for any interval, take $j_i = k + 1$, $S_i = \{i\}$, and $T_i = \emptyset$.
7. Define $\mathcal{I}'_i = \{I_1 \cup \dots \cup I_{j_i-1}, I_{j_i}, \dots, I_k\}$.
8. Run $\text{RESAMPLE}(\mathcal{A}, \mathcal{I}')$.

Note that, as part of the execution of $\text{TRIM}_\epsilon(\mathcal{A})$, a set $S_i \ni i$ and $T_i \not\ni i$ will be found for each i , which are taken to be any sets satisfying the conditions on lines 4 and 5 (or $\{i\}$ and \emptyset if no sets were found).

In summary, $\text{TRIM}_\epsilon(\mathcal{A})$ samples each constant interval for agent i , searching for appropriate sets S_i and T_i . We take I_{j_i} to be the leftmost interval for which a set S_i was found. All intervals to the left of I_{j_i} are then ironed together. Thus, regardless of the sampling outcome, I_{j_i} will be the second valuation space piece for agent i in algorithm $\bar{\mathcal{A}}_{\mathcal{I}'}$. Thus $c(S_i) \leq w_i^{\text{TRIM}_\epsilon(\mathcal{A})} + n\mu_{\max}/\sqrt{\epsilon}$ and $c(T_i) \leq n\mu_{\max}/\sqrt{\epsilon}$.

Lemma 3.3.32 *The stair compatible algorithm $\text{TRIM}_\epsilon(\mathcal{A})$ for \mathcal{A} (Definition 3.3.31) and stair thresholds $\mathbf{w}^{\text{TRIM}_\epsilon(\mathcal{A})}$ (Definition 3.3.29) satisfy $c(S_i) \leq w_i^{\text{TRIM}_\epsilon(\mathcal{A})} + n\mu_{\max}/\sqrt{\epsilon}$ and $c(T_i) \leq n\mu_{\max}/\sqrt{\epsilon}$ for all i .*

Proof: For each i , if no set satisfying the conditions on line 5 of $\text{TRIM}_\epsilon(\mathcal{A})$ was found during the sampling of any interval, then $j_i = k + 1$ and all intervals of \mathcal{A} are ironed together in \mathcal{I}' . In this case $w_i^{\text{TRIM}_\epsilon(\mathcal{A})} = \infty$, so $c(S_i) \leq w_i^{\text{TRIM}_\epsilon(\mathcal{A})}$ trivially. Otherwise, by line 5 of $\text{TRIM}_\epsilon(\mathcal{A})$, $c(S_i) \leq \min I_{j_i} + n\mu_{\max}/\sqrt{\epsilon}$. However, since all intervals to the left of I_{j_i} are ironed together in \mathcal{I}' , I_{j_i} will be the second piecewise constant interval of $\text{TRIM}_\epsilon(\mathcal{A})$, and hence $\min I_{j_i} = w_i^{\text{TRIM}_\epsilon(\mathcal{A})}$. Thus $c(S_i) \leq \min w_i^{\text{TRIM}_\epsilon(\mathcal{A})} + n\mu_{\max}/\sqrt{\epsilon}$ as required. \square

Lemma 3.3.33 $\text{TRIM}_\epsilon(\mathcal{A}) \geq \mathcal{A} - 2n\mu_{\max}\sqrt{\epsilon}$.

Proof: We claim that, with probability at least $1 - \frac{\epsilon}{2}$, if the allocation rules for $\text{TRIM}_\epsilon(\mathcal{A})$ and \mathcal{A} differ for agent i , then either $x_i(v_i) \geq 1 - (\sqrt{\epsilon} + \frac{\epsilon}{2})$ for all v_i or else $\text{TRIM}_\epsilon(\mathcal{A})$ and \mathcal{A} differ only on values v_i for which $x_i(v_i) \leq \sqrt{\epsilon} + \frac{\epsilon}{2}$. Before proving the claim, let us see how it implies the desired result. The claim implies that $\text{TRIM}_\epsilon(\mathcal{A}) \geq \mathcal{A} - (\sqrt{\epsilon} + \frac{\epsilon}{2})n\mu_{\max}$ with probability $1 - \frac{\epsilon}{2}$, as ironing together intervals whose values differ by at most $(\sqrt{\epsilon} + \frac{\epsilon}{2})$ can reduce the expected welfare by at most $(\sqrt{\epsilon} + \frac{\epsilon}{2})n\mu_{\max}$. For the remaining probability, we note that $\text{TRIM}_\epsilon(\mathcal{A}) \geq \mathcal{A} - \mathcal{A} \geq \mathcal{A} - n\mu_{\max}$ trivially, since $\mathcal{A} \leq n\mu_{\max}$, the expected welfare of allocating to all agents at no cost. Thus, over all possible outcomes of sampling, we conclude that

$$\text{TRIM}_\epsilon(\mathcal{A}) \geq \mathcal{A} - \left(\sqrt{\epsilon} + \frac{\epsilon}{2}\right)n\mu_{\max} - \frac{\epsilon}{2}n\mu_{\max} \geq \mathcal{A} - 2n\mu_{\max}\sqrt{\epsilon}$$

as required.

Let us now prove the claim. Choose some agent i and suppose that $\text{TRIM}_\epsilon(\mathcal{A})$ and \mathcal{A} differ on some interval I with $x_i(I) \geq \sqrt{\epsilon} + \frac{\epsilon}{2}$. Let I be the leftmost such interval. For the remainder of the proof we will say that a set S has *low cost for I* if $c(S) \leq \min I + n\mu_{\max}/\sqrt{\epsilon}$. Then, by the definition of \mathcal{I}'_i , it must be that no set $S \ni i$ with low cost was found during the sampling of interval I for agent i . Let us bound the probability of this event. Given $\mathbf{v} \sim \mathbf{F}$, let $B(\mathbf{v})$ be the event $[x_i(\mathbf{v}) \wedge \sum_i v_i \leq \min I + n\mu_{\max}/\sqrt{\epsilon}]$. If event $B(\mathbf{v})$ occurs for some sample \mathbf{v} , this means that \mathcal{A} returned some allocation $T \ni i$ and furthermore $\sum_i v_i \leq \min I + n\mu_{\max}/\sqrt{\epsilon}$. But note that this allocation must generate non-negative profit (otherwise it would never be allocated), and hence T must have low cost for I . Thus $B(\mathbf{v})$ is precisely the event that \mathcal{A} returns a set $T \ni i$ with low cost for I .

Consider the probability of $B(\mathbf{v})$. By Markov's inequality,

$$\Pr_{\mathbf{v}} \left[\sum_{j \neq i} v_j > n\mu_{\max}/\sqrt{\epsilon} \right] < \sqrt{\epsilon}.$$

Thus, since $v_i \geq \min I$ with probability 1 conditional on $v_i \in I$,

$$\Pr_{\mathbf{v}} \left[\sum_i v_i > \min I + n\mu_{\max}/\sqrt{\epsilon} \right] < \sqrt{\epsilon}.$$

Also, $\Pr_{\mathbf{v}}[\neg x_i(\mathbf{v}) \mid v_i \in I] = 1 - x_i(I) \leq 1 - (\sqrt{\epsilon} + \frac{\epsilon}{2})$. The union bound then implies that $\Pr_{\mathbf{v}}[\neg B(\mathbf{v})] \leq 1 - (\sqrt{\epsilon} + \frac{\epsilon}{2}) + \sqrt{\epsilon} = 1 - \frac{\epsilon}{2}$, so $\Pr_{\mathbf{v}}[B(\mathbf{v})] \geq \frac{\epsilon}{2}$.

By Chernoff-Hoeffding inequality, the probability that event B does not occur even once during $\epsilon^{-2} \log(4n/\epsilon)$ samples is at most $\frac{\epsilon}{4n}$. We conclude that the probability that no set $S \ni i$ with low cost was found during the sampling of interval I is at most $\frac{\epsilon}{4n}$. This is therefore a bound on the probability that $\text{TRIM}_{\epsilon}(\mathcal{A})$ and \mathcal{A} differ for agent i on some interval I with $x_i(I) \geq \sqrt{\epsilon} + \frac{\epsilon}{4}$. By the union bound, the probability that this occurs for *any* agent is at most $\frac{\epsilon}{4}$. Moreover, a similar argument yields that the probability that no set T with $T \not\ni i$ and $c(T) \leq n\mu_{\max}/\sqrt{\epsilon}$ was found during the sampling of an interval I' with $x_i(I') \leq 1 - (\sqrt{\epsilon} + \frac{\epsilon}{2})$, for any agent i , is at most $\frac{\epsilon}{4}$. By the union bound, we will therefore find the required sets S_i and T_i for all agents i with probability at least $1 - \frac{\epsilon}{2}$, as required. \square

We are now ready to describe the algorithm used to prove Theorem 3.3.25.

Definition 3.3.34 ($\text{MONO}_{\epsilon}(\mathcal{A})$) *Given an algorithm \mathcal{A} and any $\epsilon > 0$, the monotonization of \mathcal{A} , $\text{MONO}_{\epsilon}(\mathcal{A})$, is $\text{COMB}_{\epsilon}(\text{IRON}_{\epsilon}(\text{TRIM}_{\epsilon}(\text{DISC}_{\epsilon}(\mathcal{A}))))$.*

Lemma 3.3.35 $\text{MONO}_{\epsilon}(\mathcal{A})$ *is BIC, and $\text{MONO}_{\epsilon}(\mathcal{A}) \geq \mathcal{A} - 9kn^2\sqrt{\epsilon}\mu_{\max}$.*

Proof: For notational convenience define $\mathcal{A}' = \text{TRIM}_{\epsilon}(\text{DISC}_{\epsilon}(\mathcal{A}))$. Lemma 3.3.17 implies that $\text{IRON}_{\epsilon}(\mathcal{A}')$ is ϵ -close to a monotone algorithm, and during the construction of \mathcal{A}' we find sets S_1, \dots, S_n with $S_i \ni i$ and sets T_1, \dots, T_n with $T_i \not\ni i$. Thus $\text{COMB}_{\epsilon}(\text{IRON}_{\epsilon}(\mathcal{A}'))$ is well-defined and, by Lemma 3.3.20, is BIC.

We note that costs are not affected by our ironing techniques, and, by Lemma 3.3.33,

$$\begin{aligned} \text{IRON}_{\epsilon}(\mathcal{A}') &\geq \mathcal{A}' - n\epsilon\mu_{\max} = \text{TRIM}_{\epsilon}(\text{DISC}_{\epsilon}(\mathcal{A})) - n\epsilon\mu_{\max} \\ &\geq \text{DISC}_{\epsilon}(\mathcal{A}) - n\epsilon\mu_{\max} - 2n\mu_{\max}\sqrt{\epsilon} \geq \mathcal{A} - 5\sqrt{\epsilon}n\mu_{\max}. \end{aligned}$$

Also, $c(S_i) \leq w_i^{\mathcal{A}'} + n\mu_{\max}/\sqrt{\epsilon} \leq w_i^{\text{IRON}_\epsilon(\mathcal{A}')} + n\mu_{\max}/\sqrt{\epsilon}$ for all i by Lemma 3.3.32. Thus, by Lemma 3.3.30,

$$\begin{aligned} \text{MONO}_\epsilon(\mathcal{A}) &= \text{COMB}_\epsilon(\text{IRON}_\epsilon(\mathcal{A}')) \\ &\geq \text{IRON}_\epsilon(\mathcal{A}') - \delta(n\mu_{\max} + n\mu_{\max}/\sqrt{\epsilon}) \\ &\geq \mathcal{A} - 5n\sqrt{\epsilon}\mu_{\max} - (2(k-1)n\epsilon)2n\mu_{\max}/\sqrt{\epsilon} \\ &\geq \mathcal{A} - 9kn^2\sqrt{\epsilon}\mu_{\max}. \end{aligned}$$

□

Theorem 3.3.25 now follows immediately from Lemma 3.3.35 by considering algorithm $\text{MONO}_{\epsilon'}(\mathcal{A})$, where $\epsilon' = (\epsilon/9kn^2)^2 = \epsilon^2/81k^2n^4$. The runtime, which is dominated by sampling, is $O(kn(\epsilon')^{-2}) = \tilde{O}(n^9\epsilon^{-9}\log^5(v_{\max}/\epsilon\mu_{\max}))$.

Feasibility Settings

In feasibility settings, where costs are either 0 or infinite, the performance of algorithm $\text{MONO}_\epsilon(\mathcal{A})$ improves significantly. First, we do not need to find appropriate sets T_i by sampling; we can simply set $T_i = \emptyset$ for all i . Also, we can improve Lemma 3.3.33 as follows:

Lemma 3.3.36 *In feasibility settings, $\text{TRIM}_\epsilon(\mathcal{A}) \geq \mathcal{A} - \epsilon n\mu_{\max}$.*

Proof: Consider some agent i and interval $I \in \mathcal{I}$, and suppose $x_i(I) \geq \frac{\epsilon}{2}$. Consider the probability of finding an allocation with cost at most $\min I + n\mu_{\max}/\sqrt{\epsilon}$ when sampling for this interval. Since costs are either 0 or ∞ , this is precisely the probability of finding an allocation that includes agent i , which is $x_i(I) \geq \frac{\epsilon}{2}$. By Chernoff-Hoeffding inequality, the probability that this event does not occur even once in $\epsilon^{-2}\log(n/2\epsilon)$ samples is at most $\epsilon/2n$. We will therefore successfully find a set $S_i \ni i$ with probability at least $1 - \epsilon/2n$.

For each i , let I_{j_i} denote the leftmost interval on which $x_i(I) \geq \epsilon$. By the union bound, with probability $1 - \epsilon/2$ we will find a set $S_i \ni i$ when sampling interval I_{j_i} , for

all i . In this case, the behavior of algorithms $\text{TRIM}_\epsilon(\mathcal{A})$ and \mathcal{A} differ only on intervals I to the left of I_{j_i} , all of which satisfy $x_i(I) < \frac{\epsilon}{2}$. Thus, conditioning on an event of probability $1 - \frac{\epsilon}{2}$, $\text{TRIM}_\epsilon(\mathcal{A}) \geq \mathcal{A}(1 - \frac{\epsilon}{2}) \geq \mathcal{A} - n\mu_{\max}\epsilon$. For the remaining probability, $\frac{\epsilon}{2}$, we note that $\mathcal{A} \leq n\mu_{\max}$ trivially. We conclude that $\mathcal{A}_\epsilon \geq \mathcal{A} - \epsilon n\mu_{\max}$ unconditionally. \square

Using Lemma 3.3.36 instead of Lemma 3.3.33 in the analysis of $\text{MONO}_\epsilon(\mathcal{A})$, we find that the statement of Lemma 3.3.35 improves to show that $\text{MONO}_\epsilon(\mathcal{A}) > \mathcal{A} - 8kn^2\epsilon\mu_{\max}$ in feasibility settings. Thus, in feasibility settings, we can improve the runtime of the algorithm in Theorem 3.3.25 by taking \mathcal{A}' to be $\text{MONO}_{\epsilon'}(\mathcal{A})$ with $\epsilon' = 8kn^2\epsilon$, which has a runtime of $O(kn(\epsilon')^{-2}) = \tilde{O}(n^5\epsilon^{-5}\log^3(v_{\max}/\epsilon\mu_{\max}))$.

3.4 Limits of our Approach

3.4.1 Worst-Case Approximations

We present an example to demonstrate that our reduction from Section 3.2 does not preserve worst-case approximation ratios.

Claim 3.4.1 *There exists a 2-player single-parameter social welfare problem and algorithm \mathcal{A} such that \mathcal{A} is a worst-case c -approximation, but the ideal ironed algorithm $\bar{\mathcal{A}}$ is a d -approximation where $d > c$.*

Consider an auction of 2 objects to 2 unit-demand bidders, with the goal of optimizing social welfare. Note that the optimal solution to this problem is to always allocate one object to each bidder: the allocation $x_1 = x_2 = 1$.

Suppose that the private value of agent 1 is drawn uniformly from $\{1, 100\}$, and the private value of agent 2 is drawn uniformly from $\{10, 1000, 1001\}$. Let \mathcal{A} be an approximation algorithm whose allocation rule is described in Figure 3.4.

	$v_2 = 10$	$v_2 = 1000$	$v_2 = 1001$
$v_1 = 1$	$(x_1 = 0, x_2 = 1)$	$(x_1 = 1, x_2 = 1)$	$(x_1 = 1, x_2 = 1)$
$v_1 = 100$	$(x_1 = 1, x_2 = 0)$	$(x_1 = 0, x_2 = 1)$	$(x_1 = 0, x_2 = 1)$

Figure 3.4: The allocation rule for algorithm \mathcal{A} . The vertical axis corresponds to v_1 , the horizontal to v_2 , and the table entries are of the form “ x_1, x_2 ”. For example, if $(v_1, v_2) = (100, 10)$, then $(x_1, x_2) = (1, 0)$.

We note that \mathcal{A} is a worst-case 11/10 approximation algorithm. For example, when $v_1 = 1$ and $v_2 = 10$, the optimal solution (which allocates to both players) obtains a welfare of 11, whereas algorithm \mathcal{A} (which allocates only to player 2) obtains a welfare of 10. Also, \mathcal{A} is not BIC for agent 1, since the expected allocation is not monotone non-decreasing in his reported value: the expected allocation to player 1 when declaring value $v_1 = 1$ is 2/3, whereas his expected allocation when declaring $v_1 = 100$ is 1/3.

The ideal monotonization procedure will draw a new bid v'_1 for agent 1 uniformly from $\{1, 100\}$, and run \mathcal{A} on (v'_1, v_2) . Call this new algorithm $\bar{\mathcal{A}}$.

Suppose that $v_1 = 100$ and $v_2 = 10$. Then, with probability 1/2, $\bar{\mathcal{A}}$ will choose $v'_1 = 1$ and return allocation $(0, 1)$, and with the remaining probability it will choose $v'_1 = 100$ and return allocation $(1, 0)$. Hence, for this set of input values, the expected welfare obtained by $\bar{\mathcal{A}}$ is $\frac{100+10}{2} = 55$. Since 110 is optimal, $\bar{\mathcal{A}}$ is at best a 2-approximation algorithm, whereas \mathcal{A} is an 11/10-approximation algorithm. We conclude that ideal ironing can cause a significant decrease in worst-case approximation ratios.

3.4.2 Beyond Social Welfare

In the ideal model, our general reduction applies to any single-parameter optimization problem and converts an algorithm into a mechanism with at least the same expected social welfare. Unfortunately, this approach does not preserve other relevant objective

values, such as makespan. We illustrate this deficiency of the approach with some examples.

We consider the problem of *job scheduling on related machines* where the objective is to minimize the makespan. Here the machines are agents and each has a (privately-known) speed. The time a job takes on a machine is the product of its length and the machine's speed. The goal of the mechanism is to assign a given set of jobs with varying lengths to the machines so as to minimize the time until all machines have finished processing their jobs (i.e. the makespan).

Claim 3.4.2 *There exists a makespan minimization problem and algorithm \mathcal{A} such that the makespan of \mathcal{A} and of the ironed algorithm $\bar{\mathcal{A}}$ from Section 3.2 differ by a constant bounded away from 0.*

We consider an instance with five jobs and five machines. The job lengths, which are public knowledge, are $(j_1, j_2, j_3, j_4, j_5) = (5, 10, 5, 10, 20)$. The machine speeds are drawn probabilistically as follows. The speed of the first machine is $s_1 = 5$ with probability 1. The second machine has speed $s_2 \in \{1, 10\}$ drawn uniformly. The remaining machines have speeds $s_3 \in \{5, 10\}$ uniformly, $s_4 = 7$ with probability 1, and $s_5 = 20$ with probability 1. Recall that if a machine with speed s_i is allocated jobs with total length x_i , the resulting benefit for machine i is $-x_i/s_i$, the negative of its completion time. An algorithm for such a problem is therefore Bayesian incentive compatible if and only if the expected allocation to each agent is monotone non-decreasing as a function of its declared speed.

Suppose our problem is to minimize makespan subject to the following feasibility conditions: first, each machine must be allocated exactly one job; second, jobs 1 and 2 must be allocated to machines 1 and 2 (or vice-versa). In other words, a feasible schedule is a matching between the first two jobs and the first two machines, and a matching between the last three jobs and the last three machines.

We now describe an algorithm \mathcal{A} for this problem. The algorithm must specify a feasible allocation (i.e. a total load for each machine corresponding to a feasible schedule), for each of the four possible input profiles (that is, the four possibilities that result when $s_2 \in \{1, 10\}$ and $s_3 \in \{5, 10\}$). Our algorithm \mathcal{A} will have the following allocation rule:

input	output	makespan
$(s_1, s_2, s_3, s_4, s_5)$	$(x_1, x_2, x_3, x_4, x_5)$	
$(5, 1, 5, 7, 20)$	$(10, 5, 20, 5, 10)$	5
$(5, 1, 10, 7, 20)$	$(10, 5, 5, 10, 20)$	5
$(5, 10, 5, 7, 20)$	$(5, 10, 5, 10, 20)$	$10/7$
$(5, 10, 10, 7, 20)$	$(5, 10, 10, 5, 20)$	1

For instance, when the input profile is $(5, 1, 5, 7, 20)$, the algorithm returns allocation $(10, 5, 20, 5, 10)$. This corresponds to allocating job 2 to machine 1, job 1 to machine 2, job 5 to machine 3, job 3 to machine 4, and job 4 to machine 5. In this allocation, machine 2 has the longest finishing time, that being $x[2]/s_2 = 5/1 = 5$. The makespan of this allocation is therefore 5. We list the makespan for the other three input instances, as well. The expected makespan of \mathcal{A} is $\frac{1}{4}(5 + 5 + 10/7 + 1) \approx 3.107$.

We note that the expected allocation to player 3 is non-monotone for this algorithm. When $s_3 = 5$, machine 3 receives a load of either 20 or 5 with equal probability (recall that $s_2 \in \{1, 10\}$ uniformly) for an expected load of 12.5. However, when $s_3 = 10$, machine 3 receives a load of either 10 or 5, for an expected load of 7.5. Thus x_3 is not monotone non-decreasing in the declared speed of machine 3, so \mathcal{A} is not Bayesian incentive compatible.

The ideal monotonization procedure will draw a new speed s'_3 for machine 3 uniformly from $\{5, 10\}$, and run \mathcal{A} on $(s_1, s_2, s'_3, s_4, s_5)$. Call this new algorithm $\bar{\mathcal{A}}$. We claim that this redrawing operation can only increase the expected makespan of our algorithm. Certainly if we happen to have $s'_3 = s_3$ then the resulting makespan will be unchanged; we therefore show that whenever $s'_3 \neq s_3$ the expected makespan will increase.

Suppose $s_3 = 10$, and we take expectation over the value of s_2 . The expected makespan of \mathcal{A} when $s_3 = 10$ is 3 (the average of 5 and 1). Suppose that we instead use the output of \mathcal{A} with the modified input $s'_3 = 5$. One can then verify that the makespan when $s_2 = 1$ is 5, and when $s_2 = 10$ the makespan becomes $10/7$ (determined by machine 4, with $s_4 = 7$ and $x_4 = 10$). The expected makespan therefore increases from 3 to $45/7 \approx 6.4$ when we move from allocation $\mathcal{A}(s_1, s_2, s_3, s_4, s_5)$ to $\mathcal{A}(s_1, s_2, s'_3, s_4, s_5)$.

Next suppose $s_3 = 5$, and we again take expectation over the value of s_2 . The expected makespan is the average of 5 and $10/7$, which is $45/7 \approx 6.4$. Suppose the ironed algorithm returns the output of \mathcal{A} with the modified input value $s'_3 = 10$. In this case, when $s_2 = 1$, the makespan will be 5. When $s_2 = 10$, the makespan becomes 2 (determined by machine 3, with $s_3 = 5$ and $x_3 = 10$). The expected makespan therefore increases from $45/7$ to $\frac{1}{2}(2 + 5) = 3.5$ when we move from allocation $\mathcal{A}(s_1, s_2, s_3, s_4, s_5)$ to $\mathcal{A}(s_1, s_2, s'_3, s_4, s_5)$.

We conclude that the ironing operation increases the makespan of the resulting algorithm. Specifically, since we will have $s'_3 \neq s_3$ with probability $1/2$, the expected makespan of $\bar{\mathcal{A}}$ is $\frac{1}{2}(3.107 + \frac{1}{2}(45/7 + 3.5)) \approx 4.036$.

Note that we actually proved a stronger property of our example: *any* transformation that proceeds by applying a permutation π_3 to agent 3's reported value and then invoking \mathcal{A} must increase the expected makespan of the algorithm. We next show that this is also true of any permutation to agent 2's reported value as well. Thus, *any* modification of the algorithm that proceeds by choosing an agent i , applying a transformation π to s_i , then returning allocation $\mathcal{A}(\pi(s_i), \mathbf{s}_{-i})$ suffers a loss in approximation factor.

Suppose $s_2 = 1$, in which case the expected makespan (over choices of s_3) is 5. If we applied the output of \mathcal{A} using input $s'_2 = 10$, the expected makespan increases to 10 (determined by machine 2, with $s_2 = 1$ and $x_2 = 10$, regardless of the value of s_3). The expected makespan therefore increases from 5 to 10.

Next suppose that $s_2 = 10$, in which case the expected makespan is $17/14$, the average

of $10/7$ and 1 . If we instead apply the output of \mathcal{A} using speed $s'_2 = 5$ for machine 2, the expected makespan becomes 2 (determined by machine 1, with $s_1 = 5$ and $x_1 = 10$, regardless of the value of s_3). So again, the expected makespan increases from $17/14$ to 2 .

We conclude that all possible transformation of a single agent's value must increase the makespan of the algorithm (since any transformation is a convex combination of the value-changing operations analyzed above). It is therefore not possible, in general, to apply independent transformations the agents' values without loss.

Chapter 4

Dominant Strategy Transformations

In Chapter 3 we described a general reduction that converts an arbitrary approximation algorithm into an incentive compatible mechanism with arbitrarily small loss in expected performance, for Bayesian settings of partial information. Moreover, this reduction is black-box, requiring only oracle access to the prior type distributions and the algorithm and proceeding without knowledge of the problem’s feasibility constraints. In this chapter we consider the problem of converting approximation algorithms into incentive compatible mechanisms in non-Bayesian settings.

While Bayesian settings are ubiquitous in the economics literature, much of the existing work in theoretical computer science is focused on an alternative goal of designing computationally efficient mechanisms that are incentive compatible either *ex post* or in *dominant strategies*¹ and achieve good worst-case approximations to the optimal social welfare. In light of the success with which algorithms can be converted into mechanisms in Bayesian settings, one might ask whether there exist reductions that transform approximation algorithms into dominant strategy incentive compatible mechanisms without loss in worst-case approximation ratio. Note that this strengthens the demands of the reduc-

The results in this chapter are based upon joint work with Nicole Immorlica [54].

¹Recall that *ex post* incentive compatible mechanisms are those that are truthful in expectation, whereas dominant strategy incentive compatible mechanisms are universally truthful. See Section 2.1.

tion in two significant ways. First, it requires the stronger solution concept of dominant strategy incentive compatibility, rather than Bayesian incentive compatibility. Second, it requires that the approximation factor of the original algorithm be preserved in the worst case over all possible inputs, rather than in expectation. Can any approximation algorithm for a single-parameter social welfare problem be converted, in a black-box manner, into a dominant strategy incentive compatible mechanism without loss in worst-case approximation?

We will answer the above question in the negative by showing that any transformation that guarantees dominant strategy incentive compatibility must incur a very large loss in worst-case performance for some algorithms and problem instances. Thus, any method of converting approximation algorithms into mechanisms that are dominant strategy truthful must rely upon some extrinsically guaranteed property of the given algorithm or the problem to be solved.

While we allow our transformation to be randomized, we will require that it generate allocation rules that are *universally truthful*, meaning that each deterministic algorithm in the support of the transformed algorithm is itself dominant strategy incentive compatible. Recall that universal truthfulness is the natural notion of dominant strategy truthfulness for randomized mechanisms. We leave open the question of whether there exists a randomized transformation that guarantees the alternative notion of truthfulness in expectation, i.e. that generates randomized mechanisms that are truthful ex post, rather than in dominant strategies.

The main idea behind our impossibility result is to design a problem instance such that we can “hide” a non-monotonicity in an approximation algorithm. Roughly speaking, we consider a feasibility condition in which there are two different large allocations that can be made to certain sets of agents. When either allocation would result in high social welfare, our algorithm will choose between them in a non-monotone way. In order for a transformation to fix these non-monotonicities without impacting the performance of the

algorithm, it must switch the outcome from one to the other on some inputs; but to do so it must first determine what these allocations are, and hence find them by querying the original algorithm. However, our algorithm will have the property that, for many inputs, it is exponentially unlikely that the transformation would find both of the large allocations after polynomially many queries.

The particular allocation problem we use to establish our impossibility result has a specialized and arguably unnatural feasibility constraint. We leave as an open problem whether black-box reductions to dominant strategy truthful mechanisms are possible when we restrict ourselves to natural subclasses of social welfare problems, such as those with downward-closed feasibility constraints.

4.1 Preliminaries

Our concern in this chapter will be focused on single-parameter social welfare problems with feasibility constraints, and algorithms for those problems. We will suppose that each agent's value lies in some (publicly known) set $V_i \subseteq \mathbb{R}$ of possible types, and that the allocation algorithm must choose some allocation $\mathbf{x} \in \Gamma \subseteq \mathbb{R}^n$, where Γ denotes the set of feasible allocations², in order to maximize the social welfare $\mathbf{v} \cdot \mathbf{x}$.

Given an instance Γ of the social welfare problem, we will write $OPT_\Gamma(\mathbf{v})$ for the allocation in Γ that maximizes $\mathbf{v} \cdot \mathbf{x}$. We take the convention that the social welfare for an allocation outside Γ is $-\infty$. Given algorithm \mathcal{A} , we will write $approx_\Gamma(\mathcal{A})$ for the worst-case approximation ratio of \mathcal{A} for problem Γ . That is, $approx_\Gamma(\mathcal{A}) = \max_{\mathbf{v} \in V} \frac{OPT_\Gamma(\mathbf{v})}{\mathcal{A}(\mathbf{v})}$. Note that we have taken the convention that $approx_\Gamma(\mathcal{A}) \geq 1$ for all Γ and \mathcal{A} .

A *polynomial time transformation* \mathcal{T} is a social welfare algorithm that is given black-box access to an algorithm \mathcal{A} . We will write $\mathcal{T}(\mathcal{A}, \mathbf{v})$ for the allocation returned by \mathcal{T} on input \mathbf{v} , given that its black-box access is to algorithm \mathcal{A} . Then, for any \mathcal{A} , we can

²Note that this is a special case of the more general class of social welfare problems with cost constraints, by setting $c(\mathbf{x}) = 0$ for $\mathbf{x} \in \Gamma$ and $c(\mathbf{x}) = \infty$ otherwise.

think of $\mathcal{T}(\mathcal{A}, \cdot)$ as an algorithm for social welfare maximization; we think of this as the algorithm \mathcal{A} transformed by \mathcal{T} . We write $\mathcal{T}(\mathcal{A})$ for the allocation rule that results when \mathcal{A} is transformed by \mathcal{T} . Note that \mathcal{T} is not parameterized by Γ ; informally speaking, \mathcal{T} has no knowledge of the feasibility constraint Γ being optimized by a given algorithm \mathcal{A} .

We say that transformation \mathcal{T} is dominant strategy incentive compatible (DSIC) if, for all \mathcal{A} , $\mathcal{T}(\mathcal{A})$ is a dominant strategy incentive compatible allocation rule. Note that this incentive compatibility is independent of the problem instance Γ .

4.2 Impossibility of Lossless Transformations

In this section we show that, for any deterministic DSIC transformation \mathcal{T} , there is some allocation problem Γ and algorithm \mathcal{A} for Γ such that \mathcal{T} degrades the worst-case performance of \mathcal{A} by a polynomially large factor.

Theorem 4.2.1 *There is a constant $c > 0$ such that, for any polytime dominant strategy IC transformation \mathcal{T} , there is an algorithm \mathcal{A} and problem instance Γ such that*

$$\frac{\text{approx}_{\Gamma}(\mathcal{T}(\mathcal{A}))}{\text{approx}_{\Gamma}(\mathcal{A})} \geq n^c.$$

Our proof of Theorem 4.2.1 is constructive: we give an explicit description of the algorithm and problem instance, and we obtain constant $c = 1/20$.

The high-level idea behind our proof of Theorem 4.2.1 is as follows. We will construct an algorithm \mathcal{A} and input vectors \mathbf{v} and \mathbf{v}' such that, for each agent i in some large subset of the players, $v_i' > v_i$ but $\mathcal{A}_i(\mathbf{v}') < \mathcal{A}_i(\mathbf{v})$. This does not immediately imply that \mathcal{A} is non-truthful, but we will show that it does imply non-truthfulness under a certain feasibility condition Γ . Thus, any dominant strategy IC transformation \mathcal{T} must alter the allocation of \mathcal{A} either on input \mathbf{v} or on input \mathbf{v}' . However, we will craft our algorithm in such a way that, on input \mathbf{v} , the only allocations that the transformation will observe given polynomially many queries of \mathcal{A} will be $\mathcal{A}(\mathbf{v})$, plus allocations that have significantly worse social welfare than $\mathcal{A}(\mathbf{v})$, with high probability. Similarly, on

input \mathbf{v}' , with high probability the transformation will only observe allocation $\mathcal{A}(\mathbf{v}')$ plus allocations that have significantly worse social welfare than $\mathcal{A}(\mathbf{v}')$. Thus, in order to guarantee that it generates a truthful allocation rule, the transformation will be forced to choose an outcome with poor social welfare for either input \mathbf{v} or \mathbf{v}' , reducing the worst-case performance of the algorithm \mathcal{A} .

We now turn to a formal proof of Theorem 4.2.1 by first describing a family of feasibility constraints and algorithms, then showing that for each \mathcal{T} there is a feasibility constraint and algorithm from this family that satisfies the requirements of the theorem. This proof will be limited to deterministic transformations; in Section 4.3 we discuss extensions to randomized transformations that generate universally truthful mechanisms.

4.2.1 Construction

In the problems we consider, each private value v_i is chosen from $\{0, 1\}$. That is, we will set $V_i = \{0, 1\}$ for all $i \in [n]$. We can therefore interpret an input vector as a subset $y \subseteq [n]$, corresponding to those agents with value 1. We can therefore define $\mathcal{A}(y)$, $OPT_\Gamma(y)$, etc., for a given subset $y \subseteq [n]$. Also, for $a \geq 0$ and $y \subseteq [n]$, we will write \mathbf{x}_y^a for the allocation in which each agent $i \in y$ is allocated a , and each agent $i \notin y$ is allocated 0.

Feasible Allocations We will first define a family of feasibility constraints. Roughly speaking, we will choose some $\alpha \in (0, 1)$ and sets $S, T \subseteq [n]$ of agents. The feasible allocations will then be $\mathbf{x}_{[n]}^{\alpha^2}$, \mathbf{x}_S^1 , \mathbf{x}_T^α , and \mathbf{x}_W^α for all sufficiently small sets W . That is, we can allocate α^2 to every agent, 1 to all agents in S , α to all agents in T , or α to all agents in any set of sufficiently small size. We will also require that S and T satisfy certain properties, which essentially state that S and T are sufficiently large and have a sufficiently large intersection.

More formally, define parameters $\alpha \in (0, 1)$, $r \geq 1$, and $t \geq 1$ (which we will fix later),

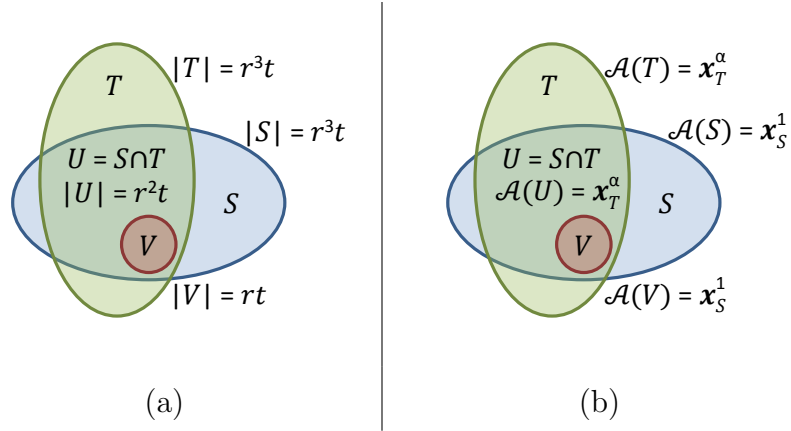


Figure 4.1: (a) Visualization of typical admissible sets of bidders V , S , and T , given size parameters r and t , and (b) the corresponding allocations of algorithm $\mathcal{A} = \mathcal{A}_{V,S,T}$.

such that $r^5t \leq n$ and $r \geq \max\{2\alpha^{-1}, 6\alpha^{-2/3}\}$. We think of t as a bound on the size of “small” sets, and we think of r as a ratio between the sizes of “small” and “large” sets. Write $\Psi = \{x_W^\alpha \mid |W| \leq t\}$ to be all allocations that allocate α to at most t agents and 0 to the remaining agents.

Suppose that V , S , and T are subsets of $[n]$. We say that the triple V, S, T is *admissible* if the following conditions hold:

1. $V \subset S \cap T$,
2. $|V| = rt$,
3. $|S \cap T| = r^2t$, and
4. $|S| = |T| = r^3t$.

In general, for a given admissible V, S , and T , we will tend to write $U = S \cap T$ for notational convenience. See Figure 4.1(a) for an illustration of the relationship between the sets in an admissible triple. For each admissible tuple V, S, T , we define a corresponding feasibility constraint

$$\Gamma_{V,S,T} = \{x_S^1, x_T^\alpha, x_{[n]}^{\alpha^2}\} \cup \Psi.$$

Note that $\Gamma_{V,S,T}$ does not depend on V ; we include set V purely for notational convenience.

The Algorithm We now define the algorithm $\mathcal{A}_{V,S,T}$ corresponding to an admissible tuple V, S, T . We think of $\mathcal{A}_{V,S,T}$ as an approximation algorithm for the social welfare problem $\Gamma_{V,S,T}$ and later show that there is no dominant strategy incentive compatible transformation of $\mathcal{A}_{V,S,T}$ without a significant loss in worst-case approximation. Given $y \subset [n]$, let $W(y)$ denote the first (up to) t agents in set y , in order of agent index. The algorithm $\mathcal{A}_{V,S,T}$ is then described as Algorithm 1.

Algorithm 1: Allocation Algorithm $\mathcal{A}_{V,S,T}$

Input: Subset $y \in [n]$ of agents with value 1

Output: An allocation $\mathbf{x} \in \Gamma_{V,S,T}$

```

1 if  $|y| < \alpha^{-1/3}t$  then
2   | return  $\mathbf{x}_{W(y)}^\alpha$ 
3 else if  $|y \cap V| > \alpha|y|$  then
4   | return  $\mathbf{x}_S^1$ 
5 else if  $|y \cap T| > \alpha^{2/3}|y|$  then
6   | return  $\mathbf{x}_T^\alpha$ 
7 else if  $|y \cap S| > \alpha^{1/3}|y|$  then
8   | return  $\mathbf{x}_S^1$ 
9 else
10  | return  $\mathbf{x}_{[n]}^{\alpha^2}$ 
11 end

```

Note that $\mathcal{A}_{V,S,T}$ always returns an outcome from $\Gamma_{V,S,T}$. For example, if we let $U = S \cap T$, we have that $\mathcal{A}(S) = \mathbf{x}_S^1$, $\mathcal{A}(T) = \mathbf{x}_T^\alpha$, $\mathcal{A}(U) = \mathbf{x}_T^\alpha$, and $\mathcal{A}(V) = \mathbf{x}_S^1$. See Figure 4.1(b) for an illustration of the relationship between these sets. The idea

underlying the algorithm is that $\mathcal{A}_{V,S,T}$ returns \mathbf{x}_S^1 if y is an approximate subset of S , and returns \mathbf{x}_T^α if y is an approximate subset of T . Otherwise, $\mathcal{A}_{V,S,T}$ allocates α^2 to all players. However, this allocation rule is ambiguous whenever y is approximately a subset of both S and T ; in this case, the algorithm will “break the tie” by returning \mathbf{x}_T^α , except when y is an approximate subset of V , in which case the algorithm returns \mathbf{x}_S^1 . Finally, in the special case that y is very small, $\mathcal{A}_{V,S,T}$ will return an allocation from Ψ .

4.2.2 Analysis

For the remainder of this section we will write $\beta = \alpha^{1/3}$ for notational convenience. We begin by bounding the approximation ratio of $\mathcal{A}_{V,S,T}$ for problem $\Gamma_{V,S,T}$, for any admissible tuple V, S, T .

Lemma 4.2.2 $approx_{\Gamma_{V,S,T}}(\mathcal{A}_{V,S,T}) \leq \alpha^{-5/3}$.

Proof: Choose any admissible tuple V, S, T and fix some $y \subseteq [n]$. We will show

$\frac{\mathcal{A}_{V,S,T}(y)}{OPT_{\Gamma_{V,S,T}}(y)} \geq \alpha^{5/3}$ by considering various cases for y .

Case 1: $|y| < \beta^{-1}t$. Then \mathcal{A} obtains social welfare $\alpha \min\{|y|, t\} > \alpha\beta|y|$, whereas $OPT(y) \leq |y|$, so $\frac{\mathcal{A}(y)}{OPT(y)} > \alpha\beta > \alpha^{5/3}$.

Case 2: $|y \cap V| \geq \beta^3|y|$. Then \mathcal{A} gets social welfare $|y \cap S| \geq |y \cap V| \geq \beta^3|y|$, whereas $OPT(y) \leq |y|$, so $\frac{\mathcal{A}(y)}{OPT(y)} \geq \beta^3 > \alpha^{5/3}$.

Case 3: $|y \cap T| \geq \beta^2|y|$, but $|y \cap V| < \beta^3|y|$. Then \mathcal{A} gets social welfare $\alpha|y \cap T| \geq \alpha\beta^2|y|$, whereas $OPT(y) \leq |y|$, so $\frac{\mathcal{A}(y)}{OPT(y)} \geq \alpha\beta^2 = \alpha^{5/3}$.

Case 4: $|y \cap S| \geq \beta|y|$. Then \mathcal{A} gets social welfare $|y \cap S| \geq \beta|y|$, whereas $OPT(y) \leq |y|$, so $\frac{\mathcal{A}(y)}{OPT(y)} \geq \beta > \alpha^{5/3}$.

Case 5: $|y \cap S| < \beta|y|$, $|y \cap T| < \beta^2|y|$, and $|y| \geq \beta^{-1}t$. Then \mathcal{A} gets social welfare $\alpha^2|y|$. The alternative \mathbf{x}_S^1 receives welfare at most $|y \cap S| < \beta|y|$ and the alternative \mathbf{x}_T^α receives

welfare at most $\alpha|y \cap T| < \alpha\beta^2|y| < \beta|y|$. Any alternative x_W^α for $|W| \leq t$ receives welfare at most $\alpha t \leq \alpha\beta|y| < \beta|y|$. We conclude $OPT(y) < \beta|y|$, so $\frac{\mathcal{A}(y)}{OPT(y)} \geq \alpha^2/\beta = \alpha^{5/3}$. \square

Suppose now that \mathcal{A}' is any algorithm for problem $\Gamma_{V,S,T}$, and suppose that \mathcal{A}' has approximation ratio better than α^2 . We will show that \mathcal{A}' is then very restricted in the allocations it can return on input $y = V$. We also show that if \mathcal{A}' is DSIC, then the allocations on inputs $y = V$ and $y = U$ are restricted further still. As any DSIC transformation of \mathcal{A} is itself an algorithm for problem $\Gamma_{V,S,T}$, these claims will later play a key role in our impossibility result.

Claim 4.2.3 *If algorithm \mathcal{A}' is such that $approx_{\Gamma_{V,S,T}}(\mathcal{A}') < \alpha^{-2}$, then $\mathcal{A}'(V) \in \{\mathbf{x}_S^1, \mathbf{x}_T^\alpha\}$.*

Proof: For $y = V$, the best possible allocation is \mathbf{x}_S^1 , for a social welfare of $|y|$. Allocation $\mathbf{x}_{[n]}^{\alpha^2}$ yields a social welfare of $\alpha^2|y|$, so \mathcal{A}' cannot return $\mathbf{x}_{[n]}^{\alpha^2}$ if $approx_{\Gamma_{V,S,T}}(\mathcal{A}') < \alpha^{-2}$. Also, for any W with $|W| \leq t$, we have $|W| \leq |V|/r \leq |y|/r$, so allocation \mathbf{x}_W^α receives social welfare at most $\alpha|y|/r < \alpha^2|y|$ (recalling that $r \geq 2\alpha^{-1}$), which is also off of the optimal by a factor of α^2 . We conclude that the required two allocations are the only ones that lead to an approximation factor better than $1/\alpha^2$, and are therefore the only allocations that can be returned by \mathcal{A}' on input V . \square

Claim 4.2.4 *Suppose \mathcal{A}' is an algorithm for problem $\Gamma_{V,S,T}$. If $\mathcal{A}'(V) = \mathbf{x}_S^1$ and $\mathcal{A}'(U) \neq \mathbf{x}_S^1$ then \mathcal{A}' is not dominant strategy incentive compatible.*

Proof: Suppose for contradiction that \mathcal{A}' is dominant strategy incentive compatible. It must therefore be that \mathcal{A}' is monotone, meaning that for all $W \subseteq [n]$ and all $i \notin W$, $\mathcal{A}'_i(W \cup \{i\}) \geq \mathcal{A}'_i(W)$.

Take any set W with $V \subseteq W \subseteq U$, $|W| = |V| + 1$. Then, on input W , \mathcal{A}' must return allocation \mathbf{x}_S^1 by monotonicity: from input V to input W a single agent's declaration increased, and that agent received an allocation of 1 on input V , so since his allocation cannot decrease it must remain 1, and the only allocation in which any agent receives 1

is \mathbf{x}_S^1 . By the same argument, \mathcal{A}' returns allocation \mathbf{x}_S^1 for all W such that $V \subseteq W \subseteq U$, and in particular for $W = U$. This contradicts the fact that \mathcal{A}' does not return \mathbf{x}_S^1 on input U . \square

In light of these claims, our strategy for proving Theorem 4.2.1 will be as follows. Given a polytime transformation \mathcal{T} , we will imagine choosing sets V , S , and T uniformly at random from all admissible tuples. We will then apply \mathcal{T} to algorithm $\mathcal{A}_{V,S,T}$ for problem $\Gamma_{V,S,T}$. What we will show that transformation \mathcal{T} is unlikely to encounter the allocation \mathbf{x}_T^α during its sampling when the input is V , over the random choice of V , S , and T . Similarly, \mathcal{T} is unlikely to encounter the allocation \mathbf{x}_S^1 during its sampling on input U . A simple application of the probabilistic method then implies that, for any transformation \mathcal{T} , there exists some choice of V , S , and T such that allocation \mathbf{x}_S^1 is not encountered during sampling on input U and allocation \mathbf{x}_T^α is not encountered during sampling on input V . Since \mathcal{T} is DSIC, Claim 4.2.4 will imply that $\mathcal{T}(\mathcal{A})$ must return an allocation outside $\{\mathbf{x}_S^1, \mathbf{x}_T^\alpha\}$ on input V . Claim 4.2.3 then implies that $\mathcal{T}(\mathcal{A})$ has approximation ratio greater than α^{-2} .

The following technical lemmas bound the likelihood that a polytime transformation will encounter the relevant allocations during its sampling, given a uniformly random choice of admissible tuples V, S, T .

Lemma 4.2.5 *Fix V and S satisfying the requirements of admissibility, and choose any y such that $|y| > \beta^{-1}t$ and $|y \cap V| < \beta^3|y|$. Then $\Pr_T[|y \cap T| > \beta^2|y|] \leq e^{-\frac{1}{r+1}(1-\beta^3)\beta^{-1}t}$, with probability taken over all choices of T that are admissible given V and S .*

Proof: Since $|y \cap V| < \beta^3|y|$, at least $(1 - \beta^3)|y|$ elements of y lie outside V . For the event $|y \cap T| > \beta^2|y|$ to occur, it must be that at least $(\beta^2 - \beta^3)|y|$ of these elements lie in T . Thus, of the elements of y chosen outside V , the fraction that fall in T must be

$$\frac{\beta^2 - \beta^3}{1 - \beta^3} = \frac{\beta^2}{1 + \beta + \beta^2} > \frac{\beta^2}{3}.$$

Any element chosen from $S - V$ has probability $\frac{|U| - |V|}{|S| - |V|} = \frac{1}{r+1}$ of being in T . Any element chosen from $[n] - S$ has probability $\frac{|T| - |U|}{[n] - |S|} \leq \frac{|U|(r-1)}{|S|(r^2-1)} < \frac{1}{r+1}$ of being in T

(recalling that $n \geq r^2|S|$). Thus, each element of y outside V has chance at most $\frac{1}{r+1}$ of being in T .

Thus, if we think of the elements of y lying outside V as $(1 - \beta^3)|y|$ independent events, each with success (i.e. lying in T) at most $\frac{1}{r+1}$, and we write X for the random variable denoting the number of such events that are successful, Chernoff bounds imply

$$\Pr \left[X > \frac{\beta^2}{3}|y|(1 - \beta^3) \right] \leq \Pr \left[X > 2\frac{1}{r+1}|y|(1 - \beta^3) \right] < e^{-\frac{1}{r+1}|y|(1-\beta^3)} < e^{-\frac{1}{r+1}(1-\beta^3)\beta^{-1}t}.$$

Note that the first inequality holds because of our assumption that $r > 6\alpha^{-2/3} = 6\beta^{-2}$. Since the event that $X > \frac{\beta^2}{3}|y|(1 - \beta^3)$ dominates the event $|y \cap T| > \beta^2|y|$, we obtain the desired result. \square

Lemma 4.2.6 *Fix T and U satisfying the requirements of admissibility (i.e. so that there are admissible tuples such that $U = S \cap T$), and choose any y such that $|y| > \beta^{-1}t$ and $|y \cap T| < \beta^2|y|$. Then $\Pr_S[|y \cap S| > \beta|y|] \leq e^{-\frac{1}{r+1}(1-\beta^2)\beta^{-1}t}$, with probability taken over all choices of S consistent with U and T .*

Proof: Since $|y \cap T| < \beta^2|y|$, at least $(1 - \beta^2)|y|$ elements of y lie outside T . For the event $|y \cap S| > \beta|y|$ to occur, it must be that at least $(\beta - \beta^2)|y|$ of these elements lie in S . Thus, of the elements of y chosen outside T , the fraction that fall in S must be at least $\frac{\beta - \beta^2}{1 - \beta^2} > \frac{\beta}{2}$.

Any element chosen from $[n] - T$ has probability $\frac{|S| - |U|}{[n] - |T|} \leq \frac{|U|(r-1)}{|T|(r^2-1)} < \frac{1}{r+1}$ of being in S (recalling that $n \geq r^2|T|$). Thus, each element of y outside T has chance at most $\frac{1}{r+1}$ of being in S .

Thus, if we think of the elements of y lying outside T as $(1 - \beta^2)|y|$ independent events, each of which occurs with probability at most $\frac{1}{r+1}$, and we write X for the random variable denoting the number of such events that are successful, Chernoff bounds imply

$$\Pr \left[X > \frac{\beta}{2}|y|(1 - \beta^2) \right] \leq \Pr \left[X > 2\frac{1}{r+1}|y|(1 - \beta^2) \right] < e^{-\frac{1}{r+1}|y|(1-\beta^2)} < e^{-\frac{1}{r+1}(1-\beta^2)\beta^{-1}t}.$$

Note that the first inequality holds because of the assumption that $r \geq 6\beta^{-2} \geq 4\beta^{-1}$. Since event $X > \frac{\beta^2}{3}|y|(1-\beta^2)$ is only more likely than the event $|y \cap S| > \beta|y|$, we obtain the desired result. \square

Lemma 4.2.7 *Fix T and U satisfying the requirements of admissibility, and choose any y such that $|y| > \beta^{-1}t$. Then $\Pr_V[|y \cap V| > \beta^3|y|] \leq e^{-\frac{1}{r}\beta^{-1}t}$.*

Proof: For the event $|y \cap V| > \beta^3|y|$ to occur, it must be that $\beta^3|y|$ of the elements in y lie in V . Thus, the fraction of elements of y that fall in V must be at least β^3 . Any element chosen from U has probability $\frac{1}{r}$ of being in V , and any element chosen from outside U has probability 0 of lying in V . Thus, each element of y has chance at most $\frac{1}{r}$ of being in V .

Thus, if we think of the elements of y as $|y|$ independent events, each with success (i.e. lying in S) at most $\frac{1}{r}$, and we write X for the random variable denoting the number of such events that are successful, Chernoff bounds imply

$$\Pr[X > \beta^3|y|] \leq \Pr\left[X > 2\frac{1}{r}|y|\right] < e^{-\frac{1}{r}|y|} < e^{-\frac{1}{r}\beta^{-1}t}.$$

Note that the first inequality holds because of the assumption that $r \geq 2\alpha^{-1} = 2\beta^{-3}$. Since event $X > \beta^3|y|$ is only more likely than the event $|y \cap V| > \beta^3|y|$, we obtain the desired result. \square

We can now set our parameters t , r , and α . We will choose $t = n^{1/5}$, $r = 2n^{3/20}$, and $\alpha = n^{-3/20}$. Note that $\beta = n^{-1/20}$. We then note that, for sufficiently large n ,

- $r^5 t \leq n$,
- $r \geq \max\{2\alpha^{-1}, 6\alpha^{-2/3}\}$, and
- $e^{-\frac{1}{r+1}\beta^{-1}(1-\beta^2)t} < e^{-\frac{1}{r}\beta^{-1}t} < e^{-n^{1/20}}$.

All of the restrictions we required of our parameters are therefore satisfied, and the probabilities from Lemmas 4.2.5, 4.2.6 and 4.2.7 are exponentially small.

Proof of Theorem 4.2.1 : For each admissible V, S, T , write $\mathcal{A}'_{V,S,T}$ for $\mathcal{T}(\mathcal{A}_{V,S,T})$. Suppose for contradiction that, for every V, S, T , $\mathcal{A}'_{V,S,T}$ has approximation ratio better (i.e. less) than α^{-2} . By Claim 4.2.3, $\mathcal{A}'_{V,S,T}(V) \in \{\mathbf{x}_T^\alpha, \mathbf{x}_S^1\}$.

For any given V, S, T , consider the sequence of queries of algorithm \mathcal{A} performed by \mathcal{T} on input $y_1 = V$. Suppose $z \subseteq [n]$ is one such query. Then if $|z| < \beta^{-1}t$, the transformation will observe an allocation in ψ . If $|z \cap V| \geq \beta^3|z|$, the transformation will observe allocation \mathbf{x}_S^1 . For any other query, Lemma 4.2.5 implies that if \mathcal{T} has no knowledge of T , then it is exponentially unlikely (over all choices of T) that \mathcal{T} will observe an allocation other than $\mathbf{x}_{[n]}^{\alpha^2}$. Thus, by the union bound, it is exponentially unlikely that any of a polynomial number of queries reveals any information about the set T , and in particular it is exponentially unlikely that \mathcal{T} will observe allocation \mathbf{x}_T^α after polynomially many queries.

Next consider the sequence of queries of algorithm \mathcal{A} performed by \mathcal{T} on input $y_2 = U$. Again, if $z \subseteq [n]$ is one such query, then if $|z| < \beta^{-1}t$ the transformation will observe an allocation in ψ . If $|z \cap U| \geq \beta^2|z|$, Lemma 4.2.7 implies that if \mathcal{T} has no knowledge of V , then it is exponentially unlikely (over all choices of V) that \mathcal{T} will observe an allocation other than \mathbf{x}_T^α . Likewise, if $|z \cap U| < \beta^2|z|$, Lemma 4.2.6 implies that it is exponentially unlikely that \mathcal{T} will observe an allocation other than $\mathbf{x}_{[n]}^{\alpha^2}$, over all choices of S . It is therefore exponentially unlikely that any query reveals any information about the set V or S beyond the identity of set U , and therefore by the union bound it is exponentially unlikely that \mathcal{T} will observe allocation \mathbf{x}_S^1 after polynomially many queries.

We conclude that, by the union bound and the probabilistic method, there is some choice of V, S, T such that $\mathcal{T}(\mathcal{A}_{V,S,T})$ does not encounter allocation \mathbf{x}_S^1 on input U and does not encounter allocation \mathbf{x}_T^α on input V . Therefore $\mathcal{A}'_{V,S,T}(U) = x_T^\alpha$ and $\mathcal{A}'_{V,S,T}(V) = x_S^1$. However, Claim 4.2.4 then implies that $\mathcal{A}'_{V,S,T}$ is not dominant strategy incentive compatible, contradicting the fact that \mathcal{T} is an DSIC transformation.

We therefore conclude that, for any DSIC transformation \mathcal{T} , there is some admissible V, S, T such that $\text{approx}_{\Gamma_{V,S,T}}(\mathcal{A}'_{V,S,T}) \geq \alpha^{-2}$. Since $\text{approx}_{\Gamma_{V,S,T}}(\mathcal{A}_{V,S,T}) = \alpha^{-5/3}$ for all admissible V, S, T , we have that $\frac{\text{approx}_{\Gamma_{V,S,T}}(\mathcal{T}(\mathcal{A}_{V,S,T}))}{\text{approx}_{\Gamma_{V,S,T}}(\mathcal{A}_{V,S,T})} \geq \alpha^{-1/3} = n^{1/20}$. \square

4.3 Randomized Transformations

We now make note of a few simple extensions of Theorem 4.2.1. First, while we proved Theorem 4.2.1 for deterministic transformations, the result extends to randomized transformations that generate universally truthful allocation rules. Indeed, suppose that transformation \mathcal{T} is such that $\mathcal{A}'_{V,S,T} = \mathcal{T}(\mathcal{A}_{V,S,T})$ is a universally truthful algorithm with $\mathbf{E}[\mathcal{A}'_{V,S,T}(V)] > \alpha^2$. Then there is a deterministic dominant strategy truthful algorithm \mathcal{A}'' in the support of $\mathcal{A}'_{V,S,T}$ such that $\mathcal{A}''_{V,S,T}(V) > \alpha^2$. This then leads to a contradiction in precisely the same manner as the proof of Theorem 4.2.1. We must therefore conclude that any such transformation satisfies $\mathbf{E}[\mathcal{A}'_{V,S,T}(V)] \leq \alpha^2$, and hence degrades the worst-case approximation factor of the algorithm $\mathcal{A}_{V,S,T}$ by a polynomially large factor.

Second, a corollary of Theorem 4.2.1 is that any deterministic transformation that converts approximation algorithms into BIC mechanisms *without access to the prior distribution* must degrade the expected performance of the approximation algorithm by a polynomially large factor for some distributions. This follows from two observations: first, if a transformation is Bayesian IC for every distribution \mathbf{F} then it must be ex post IC; and second, if a transformation degrades the expected social welfare of an algorithm by a factor of at most c for every distribution of inputs then it must degrade the worst-case approximation ratio by a factor of at most c .

Part II

Combinatorial Auction Problems

Chapter 5

Equilibria of Greedy Auctions

We now move away from single-parameter social welfare problems and consider the multi-parameter domain of combinatorial allocation problems. In such a problem, the goal is to assign m objects to n agents in order to maximize the social welfare, subject to arbitrary downward-closed feasibility constraints. This class includes combinatorial auctions, multi-unit combinatorial auctions, unsplittable flow problems, and many others.

For many combinatorial allocation problems, straightforward deterministic greedy algorithms meet or approach the best-known approximation factors subject to computational constraints [64, 71, 18, 7]. These greedy methods tend to be very efficient to implement and easy for bidders to understand, which are desirable properties in auctions. Unfortunately, such algorithms tend not to be truthful in general [64]. One might be tempted to assume there is no recourse but to abandon these methods in favour of other, potentially more complex, mechanisms. However, we will take a different route in this chapter: we will study the game-theoretic properties of greedy algorithms in more detail, and argue that they may still be viable candidates for mechanism allocation rules.

Our approach will be to study mechanisms for combinatorial allocation problems that yield good approximations at equilibria of agent behaviour. Performance at equilibrium

The results in this chapter are based upon joint work with Allan Borodin [67].

has been studied extensively in the algorithmic game theory literature as the *price of anarchy* of a given game: the ratio between the optimal outcome and the worst-case outcome at any equilibrium [79]. Our approach is to apply these well-studied equilibrium notions directly to the field of algorithmic mechanism design, with the goal of converting algorithms into mechanisms with comparable performance at every equilibrium.

As with the solution concept of incentive compatibility, we can envision both full-information and partial-information variants of this mechanism design problem. In the full information setting, the natural equilibrium concept is (pure or mixed) Nash equilibrium. That is, we suppose that each agent knows the types of all other agents, and that the agents use this knowledge to rationally select strategies that are in equilibrium. Thus, a bound on performance at equilibrium under this model leverages the fact that agents have full knowledge of all types. This full-information assumption may be appropriate in some settings, such as when an auction is repeated multiple times so that agents have an opportunity to learn each others' types. However, in one-shot auctions, this assumption can be quite unreasonable. It is for this reason that we focus our attention primarily on the incomplete information setting, where the appropriate equilibrium concept is Bayes-Nash equilibrium. That is, we assume only that agents have some partial information about their opponents in the form of type distributions, and that they apply strategies at equilibrium given this partial knowledge. We pose the question: can a given black-box approximation algorithm be converted into a mechanism that approximately preserves its approximation ratio at every Bayes-Nash Equilibrium? We show that for a broad class of greedy algorithms, the answer is *yes*.

A few points require clarification regarding our solution concept. The concept of Bayes-Nash equilibrium requires that agents' types are drawn from commonly-known distributions. However, our mechanisms will not depend on the actual distributions themselves: we will present a single (deterministic) mechanism that works for *every* distribution. In economic terms, the mechanisms we consider are *detail-free*. Thus, while

we assume the existence of type distributions in order to model rational agent behaviour, our mechanism need not be aware of these distributions.

Recall that every Nash equilibrium is also a Bayes-Nash equilibrium, so our mechanisms will also preserve approximation ratios at every (mixed or pure) Nash equilibrium of the full information game. This is precisely the concept of *price of anarchy* from the algorithmic game theory literature. Following Christodoulou et al. [23], we extend this notion to the *Bayesian price of anarchy*, which is the worst-case approximation attained at any Bayes-Nash equilibrium of the mechanism. Indeed, our motivating question can be rephrased as asking whether every c -approximation algorithm can be implemented as a mechanism with Bayesian price of anarchy at most $c(1 + o(1))$.

As is standard, our main bounds on the Bayesian price of anarchy will assume that agent types are distributed independently. However, we show that a weaker bound of $O(c)$ holds when agent types are drawn from an arbitrary distribution over the space of all type profiles. Thus, even if agent types are arbitrarily correlated, our mechanisms yield performance nearly matching that of the original algorithm, asymptotically, at equilibrium.

Dominant strategy truthfulness of an approximation mechanism is a conceptually stronger solution concept than requiring that a mechanism approximate the optimal social welfare at every equilibrium. However, as noted elsewhere [23], the NE and BNE solution concepts are not, strictly speaking, relaxations of dominant strategy truthfulness. There exist truthful mechanisms whose approximation ratios are not preserved at all Nash equilibria, such as the famous Vickrey auction.

While our mechanisms will be implementable in polynomial time (assuming an efficient implementation of the original approximation algorithm), we do not argue that a (Bayesian or non-Bayesian) equilibrium of the underlying game can necessarily be found in polynomial time. We do not explicitly model the manner by which agents reach equilibrium, or impose upon the agents any computational constraints whatsoever. We

simply assume, as is common in the economics literature, that agents will be able to find equilibria (or approximate equilibria) of the mechanism. This is an admittedly strong assumption, and in Chapter 6 we will extend our analysis to alternative solution concepts that explicitly model agent behaviour.

Our approach in this chapter is to convert a c -approximate algorithm into a mechanism with Bayesian price of anarchy approaching c . However, we note that if agents are not computationally bounded, it may be possible to design mechanisms that perform better, at equilibrium, than any polynomial-time algorithm. Indeed, it may be possible to create a mechanism that generates optimal outcomes at equilibrium for computationally hard problems. For example, a (non-direct-revelation) mechanism may directly ask agents to suggest outcomes that optimize the social welfare. However, we would argue that such solutions are unnatural and contrary to the spirit of the mechanism design problem, in that we wish to implement mechanisms that perform well *despite* the power of agent rationality, rather than *because* of an ability to exploit this power. Additionally, we view our motivating problem not as designing an arbitrary mechanism that achieves a certain approximation bound at equilibrium, but rather as studying the properties of simple, natural mechanisms in order to argue that they will generate reasonable outcomes under rational manipulation.

Finally, our analysis shares some similarity with the properties of equilibria of smooth games, introduced independently by Roughgarden [82]. We show, however, that our analysis is distinct in that our combinatorial auction mechanisms are not smooth games.

Our Results We will describe the main results and theorem statements of this chapter informally; the formal statements appear in their corresponding sections. We begin by considering pure equilibria in a full-information setting. We show that if a greedy algorithm is paired with a first-price payment scheme (i.e. each agent pays his declared value for the set he receives), the resulting auction preserves the original algorithm's

approximation ratio at every pure Nash equilibrium.

Theorem 5.3.2 *Any greedy c -approximation algorithm for a combinatorial allocation problem can be implemented as a first-price mechanism that achieves a c approximation at every pure Nash equilibrium.*

The above result is marred by two issues. First, as discussed above, the full-information model is arguably unnatural for one-shot combinatorial auctions. Second, we demonstrate that there are instances in which no pure Nash equilibria exist. To address the latter, we extend our analysis to mixed equilibria; to address the former issue, we extend to Bayes-Nash equilibria. We show that the first-price mechanism yields a Bayesian Price of Anarchy that matches the approximation ratio of the original greedy algorithm, subject to an exponentially small loss.

Theorem 5.3.4 *Any greedy c -approximation algorithm for a combinatorial allocation problem can be implemented as a first-price mechanism with Bayesian price of anarchy at most $c + O(c^2/e^c)$.*

We also show that the loss in approximation factor is not an artifact of the analysis: there exist examples in which the first-price mechanism can have a $(c + \Omega(c^2/e^{4c}))$ approximation at equilibrium.

Note that we prove our theorem for all *mixed* Bayes-Nash equilibria, meaning that the approximation ratio holds even if agents can choose distributions over strategies. This is somewhat non-standard, as Bayesian equilibria are usually considered to include only pure strategies. Nevertheless, our result is more general than a consideration of only pure-strategy Bayes-Nash equilibria, and also generalizes the (standard) notion of mixed (non-Bayesian) Nash equilibria.

One issue with first-price mechanisms is that agents may find it difficult to determine rational strategies, since agents must strategically choose their declared values for sets in order to optimize the price they pay for whichever set of objects they acquire. As an

alternative, we also consider a mechanism that charges *critical prices*. Such a mechanism has the property that if an agent wins a set S , then the amount he pays for that set does not depend on his declaration. However, as has been noted elsewhere [23, 65], mechanisms of this form can suffer from unnatural problems at equilibrium: an agent may have incentive to greatly over-represent his values, hoping that no other agent makes large bids. Indeed, we construct examples in which an agent might have a strict preference for following such overbidding strategies in the presence of uncertainty. This possibility of overbidding can result in equilibria with poor social welfare. However, these bidding strategies are inherently risky: depending on the bids of other agents, an overbidding agent may end up with negative utility. If we can assume that agents do not participate in this risky behaviour, we can once again bound the approximation ratio we attain at equilibrium.

Theorem 5.4.2 *Under the assumption that agents never declare more than their true values on any set, any greedy c -approximation algorithm for a combinatorial allocation problem can be implemented as a critical price mechanism that achieves a $(c + 1)$ approximation at every Bayesian Nash equilibrium.*

We show that the loss in approximation factor from c to $c + 1$ is essential as c grows large; for every $c \geq 1$ there exists an instance in which a greedy algorithm has approximation factor c but the associated critical-price mechanism has price of anarchy $c + 1 - O(\frac{1}{c})$. We can, however, tighten our approximation ratios to c for certain special cases. This includes algorithms that are $(c - 1)$ -approximate when agents are single-minded, as well as algorithms that are symmetric with respect to the agents and objects (i.e. do not depend on agent or object labels). We discuss these improvements in Section 5.6. We also note that if agents apply strategies that form a γ -approximate Bayes-Nash equilibrium for some $\gamma \geq 1$, our bound on the approximation factor degrades gracefully from $c + 1$ to $c + \gamma$.

We also provide an alternative analysis of the first-price mechanism that does not

give as tight a bound on the price of anarchy, but applies even if agent types are not independently distributed. We can therefore bound the Bayesian price of anarchy of a greedy approximation mechanism even for arbitrarily correlated agent types. This extension requires that we restrict our attention to the class of non-adaptive greedy algorithms.

Theorem 5.3.9 *Any non-adaptive greedy c -approximation algorithm for a combinatorial allocation problem can be implemented as a first-price mechanism that achieves a $4c$ approximation at every Bayesian Nash equilibrium for every (possibly correlated) type distribution.*

All of the above results make use of a property of greedy algorithms that we term *strong loser-independence*. Roughly speaking, an algorithm is strongly loser-independent if any two declaration profiles \mathbf{d} and \mathbf{d}' that differ only on “losing bids” (i.e. agents’ declared values for sets that are not allocated to them) will result in the same allocation. This definition is motivated by a different notion of loser-independence for single-parameter problems introduced by Chekuri and Gamzu [21].

Finally, we show how to extend our results for greedy algorithms to a combination of a greedy allocation rule with a rule that allocates all objects to a single bidder. These combinations are a useful tool for constructing algorithms that can be implemented efficiently, such as for the general combinatorial auction problem [71]. Such algorithms are not necessarily strongly loser-independent, so slightly different techniques are required to bound their performance at equilibrium.

5.1 Preliminaries

5.1.1 Feasible Allocation Problems

We consider a setting in which there are n agents and a set M of m objects. An *allocation* to agent i is a subset $x_i \subseteq M$. A *valuation function* $v : 2^M \rightarrow \mathbb{R}$ assigns a value to each allocation. We assume that valuation functions are monotone, meaning $v(S) \leq v(T)$ for all $S \subseteq T \subseteq M$, and normalized so that $v(\emptyset) = 0$. A valuation function v is *single-minded* if there exists a set $S \subseteq M$ and a value $x \geq 0$ such that for all $T \subseteq M$, $v(T) = x$ if $S \subseteq T$ and 0 otherwise. The *zero valuation* sets $v(S) = 0$ for all $S \subseteq M$; we will represent this special valuation by \emptyset .

The goal in our social welfare maximization problems is to choose an allocation for each agent in order to maximize the sum of agent values. We assume the presence of a feasibility constraint that limits the set of allowable allocation profiles. We further assume in combinatorial allocation problems that this feasibility constraint is *separable*, meaning that if \mathbf{x} is feasible then $(\emptyset, \mathbf{x}_{-i})$ is also feasible for all i . Note that separability is a weaker assumption than the standard downward-closure property of packing problems, which would stipulate that if \mathbf{x} is feasible then (y_i, \mathbf{x}_{-i}) is also feasible for all $y_i \subseteq x_i$.

5.1.2 Greedy Allocation Rules

We describe a special type of allocation rule, which we will refer to as a *greedy allocation rule*. These are motivated by the monotone greedy algorithms of Mu'alem and Nisan [71], extended to be adaptive. We begin with some definitions. A *partial allocation profile* for agents $N \subseteq [n]$ is an allocation profile \mathbf{x} with $x_i = \emptyset$ for all $i \notin N$. A partial allocation profile is *feasible* if there is some feasible allocation profile that extends it. Given a partial allocation profile \mathbf{x} for subset N , some $i \notin N$, and allocation $y \subseteq M$, we say y is a *feasible allocation for i given N* if the partial allocation remains feasible when we set $x_i = y$.

A *priority function* is a function $r : [n] \times 2^M \times \mathbb{R} \rightarrow \mathbb{R}$. We think of $r(i, S, v)$ as the

priority of allocating $S \subseteq M$ to player i when $v_i(S) = v$. We require r to be monotone non-decreasing in v and monotone non-increasing in S with respect to set inclusion.

We consider two types of greedy allocation algorithms. A *non-adaptive greedy allocation algorithm* is an allocation algorithm of the following form:

1. Fix a monotone priority function r . Let $N = [n]$.
2. Repeat until $N = \emptyset$:
3. Choose $i \in N$ and $S \subseteq M$ that maximizes $r(i, S, d_i(S))$ over all feasible allocations S
4. Set $X_i = S$; remove player i from N
5. return X_1, \dots, X_n

We assume that ties in step 3 are broken in an arbitrary but fixed manner. A non-adaptive algorithm fixes a single priority function that is used throughout its execution. By contrast, an *adaptive greedy allocation algorithm* can change its priority function on each iteration, depending on the partial allocation formed on the previous iterations.

Some examples of greedy algorithms for various combinatorial allocation problems are described in Section 5.7.

5.1.3 Mechanisms and Payment Methods

Given an allocation rule \mathcal{A} , we will write $\mathcal{M}_1(\mathcal{A})$ to denote the mechanism that applies allocation rule \mathcal{A} and the first-price payment scheme. We call this the *first-price mechanism for \mathcal{A}* . The *critical-price mechanism for \mathcal{A}* , $\mathcal{M}_{crit}(\mathcal{A})$, instead applies the critical price payment scheme.

5.1.4 Smoothness

Bounding the price of anarchy for a given game is a common task in algorithmic game theory. Recently, Roughgarden [82] showed that many existing price of anarchy results can be expressed as a general *smoothness* argument, which furthermore results in matching bounds on the (pure) price of anarchy and coarse correlated price of anarchy. In this section we demonstrate that this smoothness argument does not apply directly to our setting of social welfare maximization in greedy combinatorial auction mechanisms.

Let us first recall the original definition of smoothness. In a payoff maximization game, each of n agents chooses a strategy s_i from strategy space Γ_i . Each agent then obtains a payoff (i.e. utility) $u_i(\mathbf{s})$ which depends on all of the strategies chosen.

Given $\lambda \geq 0$ and $\mu \leq 1$, a payoff-maximization game¹ is (λ, μ) -smooth if, for all strategy profiles \mathbf{s} and \mathbf{s}' ,

$$\sum_i u_i(s'_i, \mathbf{s}_{-i}) \geq \lambda \sum_i u_i(\mathbf{s}') + \mu \sum_i u_i(\mathbf{s}).$$

Note that a game can be (λ, μ) -smooth only if $\lambda + \mu \leq 1$ (consider the case $\mathbf{s}' = \mathbf{s}$). Roughly speaking, the smoothness condition guarantees that if strategy profile \mathbf{s}' yields a higher total welfare than \mathbf{s} , then this improvement is reflected in the average improvement to each individual player's change in utility when changing their own strategy unilaterally from \mathbf{s} to \mathbf{s}' . Roughgarden showed that any (λ, μ) -smooth payoff-maximization game with $\lambda > 0$ and $\mu < 1$ has price of anarchy equal to $\frac{1-\mu}{\lambda}$, and that this bound extends to a range of other equilibrium notions, including the coarse correlated price of anarchy [82].

This notion of smoothness can be extended to the realm of mechanism design in a natural way. Given a vector of agent types \mathbf{v} , mechanism \mathcal{M} induces a game in which each agent is a player and the strategies are type declarations, \mathbf{d} . We also include an extra player, with only a single available strategy, to represent the mechanism; the utility

¹The original formulation of smoothness applied to cost minimization games, but is readily extended to payoff-maximization domains.

of this player will be precisely the sum of the payments of the other players. We include this extra player so that the sum of player utilities is equal to the social welfare of the mechanism's outcome.

We then say that mechanism \mathcal{M} is (λ, μ) -smooth if, for all \mathbf{v} and all strategy profiles \mathbf{d} and \mathbf{d}' ,

$$\sum_i p_i(\mathbf{d}) + \sum_i u_i(d_i', \mathbf{d}_{-i}) \geq \lambda SW(\mathbf{x}(\mathbf{d}'), \mathbf{v}) + \mu SW(\mathbf{x}(\mathbf{d}), \mathbf{v}).$$

However, as we now show, our greedy mechanisms are not smooth games for any $\lambda > 0$ and $\mu \leq 1$, and hence we cannot use smoothness to bound its price of anarchy.

Example 5.1.1 *Consider a combinatorial auction with two objects, $\{a, b\}$, and two players. Our mechanism will greedily accept bids by value and charge any individually rational pricing scheme (i.e. no agent is charged more than her declared value for her winnings).*

Choose $r > 1$, and suppose the types of the players are given by $v_1(a) = v_1(\{a, b\}) = r$, $v_1(b) = 1$ and $v_2(b) = v_2(\{a, b\}) = r$, $v_2(a) = 1$. In strategy profile \mathbf{d} , agent 1 declares a value of 1 single-mindedly for item $\{b\}$, and agent 2 declares a value of 1 single-mindedly for item $\{a\}$. In strategy profile \mathbf{d}' , agent 1 declares a value of $1/2$ single-mindedly for $\{a\}$, and agent 2 declares a value of $1/2$ single-minded for $\{b\}$.

In this example, we have that $SW(\mathbf{x}(\mathbf{d}), \mathbf{v}) = 2$ (with $x_1 = \{b\}$ and $x_2 = \{a\}$) and $SW(\mathbf{x}(\mathbf{d}'), \mathbf{v}) = 2r$ (with $x_1 = \{a\}$ and $x_2 = \{b\}$). However, if agent 1 bids d_1 and agent 2 bids d_2' , then both agents are bidding single-mindedly for item b ; agent 1 will win it because he has the larger bid, and agent 2 will win \emptyset for a utility of 0. Similarly, if agent 1 bids d_1' and agent 2 bids d_2 then the utility of agent 2 will be 0. Moreover, the sum of payments under strategy profile \mathbf{d} is at most 2 by individual rationality.

Thus, if our mechanism is (λ, μ) -smooth for $\lambda \geq 0$ and $\mu \leq 1$, it must be that, for all $r > 1$, we have

$$2 + 0 \geq \lambda 2r + \mu 2.$$

However, this is true for all $r > 1$ only if $\lambda = 0$. We conclude that our mechanism is not (λ, μ) -smooth for any $\lambda > 0$.

5.2 Strong Loser-Independence

Chekuri and Gamzu [21] introduced a property known as loser-independence for combinatorial allocation algorithms in single-parameter domains. They define an algorithm for a combinatorial allocation problem to be *loser-independent* if, whenever $\mathcal{A}_i(d_i, \mathbf{d}_{-i}) = \mathcal{A}_i(d'_i, \mathbf{d}_{-i}) = \emptyset$ for some i , \mathbf{d}_{-i} , d_i , and d'_i , then it must be that $\mathcal{A}(d_i, \mathbf{d}_{-i}) = \mathcal{A}(d'_i, \mathbf{d}_{-i})$. That is, if a “losing” agent (i.e. an agent who is allocated no items) modifies his declaration in such a way that he still receives no items, this cannot affect the outcome of algorithm \mathcal{A} . In our results we will make use of a stronger property of greedy algorithms, which we call strong loser-independence.

Definition 5.2.1 *An allocation rule \mathcal{A} is strongly loser-independent if, whenever \mathbf{d} and \mathbf{d}' satisfy $\mathcal{A}(\mathbf{d}) \neq \mathcal{A}(\mathbf{d}')$, there exists an agent i and set S such that $d_i(S) \neq d'_i(S)$ and either $\mathcal{A}_i(\mathbf{d}) = S$ or $\mathcal{A}_i(d'_i, \mathbf{d}_{-i}) = S$.*

Roughly speaking, if \mathcal{A} is a strongly loser-independent algorithm, then whenever a valuation profile changes from \mathbf{d} to \mathbf{d}' via modifications to “losing bids” (i.e. agents’ declared values for sets that are not allocated to them), algorithm \mathcal{A} will return the same outcome on inputs \mathbf{d} and \mathbf{d}' . We note that our definition requires that either $\mathcal{A}_i(\mathbf{d}) = S$ or $\mathcal{A}_i(d'_i, \mathbf{d}_{-i}) = S$, rather than $\mathcal{A}_i(\mathbf{d}') = S$. The intuition behind this choice is that we wish to think of “losing bids” as being losers with respect to the original declaration profile \mathbf{d} .

The strong loser-independence property strengthens Chekuri and Gamzu’s definition of loser-independence in two ways. First, we extend from single-parameter settings to multiple-parameter settings by considering losing bids rather than losing agents. Second,

we require that the algorithm outcome be unaffected if multiple agents simultaneously modify losing bids.

It is clear from the definitions that all strongly loser-independent algorithms are loser-independent (i.e. by considering the case when \mathbf{d} and \mathbf{d}' differ only on the declaration of a single agent). However, not all loser-independent algorithms are strongly loser-independent, even in single-minded domains. For example, consider the combinatorial auction problem and suppose that \mathcal{A} is an algorithm that optimizes social welfare exactly and breaks ties consistently. Then \mathcal{A} is loser-independent, since a losing agent's bid does not affect the optimal allocation. However, \mathcal{A} is not strongly loser-independent, as the following instance shows. Consider an auction of two items $\{a, b\}$ to three bidders. If the (single-minded) bidder declarations are $d_1(\{a, b\}) = 10$, $d_2(\{a\}) = 3$, and $d_3(\{b\}) = 3$, then the outcome is that agent 1 wins his desired set. On the other hand, if the bidder declarations are given by \mathbf{d}' where $d_1'(\{a, b\}) = 10$, $d_2'(\{a\}) = 6$, and $d_3'(\{b\}) = 6$, then the outcome changes: agents 2 and 3 win their desired sets. However, \mathbf{d} and \mathbf{d}' do not differ in their declarations for any sets allocated by $\mathcal{A}(\mathbf{d})$, $\mathcal{A}(d_1', \mathbf{d}_{-i})$, $\mathcal{A}(d_2', \mathbf{d}_{-i})$, or $\mathcal{A}(d_3', \mathbf{d}_{-i})$, as agent 1 wins his desired set in each of these four cases. This contradicts the definition of strong loser-independence.

As we now show, all greedy algorithms satisfy the strong loser-independence property.

Lemma 5.2.2 *Every adaptive greedy algorithm is strongly loser-independent.*

Proof: Let \mathcal{A} be an adaptive greedy allocation rule, and choose any \mathbf{d} and \mathbf{d}' such that $\mathcal{A}(\mathbf{d}) \neq \mathcal{A}(\mathbf{d}')$. We will show that there exists some i and S such that $d_i(S) \neq d_i'(S)$ and either $\mathcal{A}_i(\mathbf{d}) = S$ or $\mathcal{A}_i(d_i', \mathbf{d}_{-i}) = S$.

Recall the definition of an adaptive greedy algorithm, and consider the iterations of \mathcal{A} on inputs \mathbf{d} and \mathbf{d}' . Let k be the first iteration in which the allocation of \mathcal{A} differs on these two inputs. Suppose that \mathcal{A} allocates set U to agent ℓ on iteration k when the input is \mathbf{d} , and allocates T to agent j on iteration k when the input is \mathbf{d}' .

For each iteration $q < k$, write i_q for the agent allocated to by \mathcal{A} (on either input profile) and S_q for the set allocated to i_q . Note that if $d_{i_q}(S_q) \neq d_{i_q}'(S_q)$ for any $q < k$ then we have the desired result with $i = i_q$ and $S = S_q$. We can therefore assume that $d_{i_q}(S_q) = d_{i_q}'(S_q)$ for all $q < k$. This implies that the bids resolved by \mathcal{A} are identical on all iterations preceding k on inputs \mathbf{d} and \mathbf{d}' , and therefore the ranking function used on each iteration up to k must be identical for inputs \mathbf{d} and \mathbf{d}' . Write r_q for the ranking function used on iteration q for each $q \leq k$. Thus, since the allocation on iteration k changed from choosing set U for agent ℓ to choosing set T for agent j , it must be that either $r_k(\ell, U, d_\ell(U)) \neq r_k(\ell, U, d_\ell'(U))$ or $r_k(j, T, d_j(T)) \neq r_k(j, T, d_j'(T))$. This implies that either $d_\ell(U) \neq d_\ell'(U)$ or $d_j(T) \neq d_j'(T)$.

If $d_\ell(U) \neq d_\ell'(U)$ then we have the desired result with $i = \ell$ and $S = U$, since $\mathcal{A}_\ell(\mathbf{d}) = U$. We can therefore assume that $d_\ell(U) = d_\ell'(U)$ and $d_j(T) \neq d_j'(T)$. Consider now the behaviour of algorithm \mathcal{A} on input (d_j', \mathbf{d}_{-i}) . We claim that $\mathcal{A}_j(d_j', \mathbf{d}_{-i}) = T$. Note that this implies the desired result with $i = j$ and $S = T$. To prove the claim, recall that $d_{i_q}(S_q) = d_{i_q}'(S_q)$ for all $q < k$. Thus, for each $q < k$ and each feasible set S that could be allocated to agent j on iteration q ,

$$r_q(i_q, S_q, d_{i_q}(S_q)) = r_q(i_q, S_q, d_{i_q}'(S_q)) > r_q(j, S, d_j'(S))$$

since \mathcal{A} allocates S_q to i_q on input \mathbf{d}' . We conclude that, on input (d_j', \mathbf{d}_{-i}) , \mathcal{A} allocates S_q to agent i_q on each iteration $q < k$. On iteration k , we have $r_k(j, T, d_j'(T)) > r_k(j, T', d_j'(T'))$ for any feasible $T' \neq T$ (since \mathcal{A} allocates T to j on input \mathbf{d}') and

$$r_k(j, T, d_j'(T)) > r_k(\ell, U, d_\ell'(U)) = r_k(\ell, U, d_\ell(U)) \geq r_k(i, S, d_i(S))$$

for any feasible $i \neq j$, due to our assumption that $d_\ell'(U) = d_\ell(U)$ and the fact that \mathcal{A} allocates U to ℓ on iteration k for input \mathbf{d} . We therefore conclude that \mathcal{A} allocates set T to agent j on iteration k , and hence $\mathcal{A}_j(d_j', \mathbf{d}_{-i}) = T$ as required. \square

We next explore an implication of a strongly loser-independent algorithm \mathcal{A} being a worst-case c -approximation. If \mathcal{A} is a c -approximate algorithm, then (on any input) the

sum of the declared values for its output profile approximates the sum of the declared values for the optimal allocation. We now show that it also approximates the sum of the critical prices of the optimal allocation profile.

Lemma 5.2.3 *If \mathcal{A} is a c -approximate strongly loser-independent algorithm, then for any type profile \mathbf{v} and allocation profile \mathbf{y} , $\sum_{i \in [n]} v_i(\mathcal{A}_i(\mathbf{v})) \geq \frac{1}{c} \sum_{i \in [n]} \theta_i(y_i, \mathbf{v}_{-i})$.*

Proof: Choose any $\epsilon > 0$. For all i , let v_i' be the single-minded declaration for set y_i at value $\theta_i(y_i, \mathbf{v}_{-i}) - \epsilon$. Let v_i^* be the pointwise maximum of v_i' and v_i . That is, for all $S \subseteq M$, $v_i^*(S) = \max\{v_i(S), v_i'(S)\}$. By definition of critical prices, we have that $\mathcal{A}_i(v_i^*, \mathbf{v}_{-i}) = \mathcal{A}_i(\mathbf{v})$ for all i , and furthermore $v_i^*(\mathcal{A}_i(\mathbf{v})) = v_i(\mathcal{A}_i(\mathbf{v}))$. Since \mathcal{A} is strongly loser-independent, we must therefore have $\mathcal{A}(\mathbf{v}) = \mathcal{A}(\mathbf{v}^*)$. Since \mathcal{A} is a c -approximation, we conclude that $SW(\mathbf{x}(\mathbf{v}), \mathbf{v}) = SW(\mathbf{x}(\mathbf{v}^*), \mathbf{v}^*) \geq \frac{1}{c} SW(\mathbf{y}, \mathbf{v}^*) \geq \frac{1}{c} \sum_{i \in [n]} \theta_i(y_i, \mathbf{v}_{-i}) - n\epsilon$. The result follows by taking the limit as $\epsilon \rightarrow 0$. \square

5.3 First-Price Mechanisms

In this section we demonstrate how to implement a greedy allocation rule so that, at any Bayes-Nash equilibrium of the resulting mechanism, the approximation ratio of the greedy rule is nearly preserved. We will be studying the performance of the first-price mechanism $\mathcal{M}_1(\mathcal{A})$ at equilibrium.

Our first step will be to show that the utility-maximizing declaration of an agent never involves overbidding on a set that he may possibly be allocated. This will allow us to assume that agents avoid overbidding strategies. It may appear at first glance that any strategy that recommends overbidding on sets is obviously dominated, since winning any bid larger than one's true value leads to negative utility. However, we must also show that an agent cannot find it advantageous to overbid on some set S in order to affect his probability of winning some other set T . We will demonstrate that such situations cannot occur when allocations are chosen by a greedy algorithm.

We say that strategy $b_i(\cdot)$ is an *overbidding strategy* if there exists type v_i and set S such that $(b_i(v_i))(S) > v_i(S)$. Given strategy b_i , we will write \underline{b}_i for the strategy that proceeds as b_i , but places an upper bound of $v_i(S)$ on the bid for any set S . That is, \underline{b}_i maps type v_i to declaration d_i where $d_i(S) = \min\{(b_i(v_i))(S), v_i(S)\}$. Note that $\underline{b}_i = b_i$ precisely if b_i does not allow overbidding.

We now show that any strategy b_i that overbids on a set that could potentially be won is strictly dominated by strategy \underline{b}_i .

Lemma 5.3.1 *For any adaptive greedy algorithm \mathcal{A} , every strategy b_i for mechanism $\mathcal{M}_1(\mathcal{A})$ is weakly dominated by \underline{b}_i . Moreover, this domination is strict if there exist v_i , \mathbf{d}_{-i} , and S such that $(b_i(v_i))(S) > v_i(S)$ and $\mathcal{A}_i(b_i(v_i), \mathbf{d}_{-i}) = S$.*

Proof: Choose type v_i for agent i and let $d_i = b_i(v_i)$. Let $d_i' = \underline{b}_i(v_i)$, so $d_i'(S) = \min\{d_i(S), v_i(S)\}$ for all $S \subseteq M$. Note that d_i' satisfies monotonicity.

Fix any possible declaration \mathbf{d}_{-i} by the other agents. We claim that $u_i(d_i', \mathbf{d}_{-i}) \geq u_i(d_i, \mathbf{d}_{-i})$. Let $S_i = \mathcal{A}_i(d_i', \mathbf{d}_{-i})$ and $T_i = \mathcal{A}_i(d_i, \mathbf{d}_{-i})$. If $d_i(T_i) \geq v_i(T_i)$ then $u_i(d_i, \mathbf{d}_{-i}) \leq 0 \leq u_i(d_i', \mathbf{d}_{-i})$ as claimed. If, on the other hand, $d_i(T_i) < v_i(T_i)$, then $d_i'(T_i) = d_i(T_i)$ by definition. From the definitions of S_i and T_i , and of an adaptive greedy algorithm, it must be that (on some iteration of \mathcal{A}) S_i has a higher priority than T_i under declaration d_i' , but T_i has a higher priority than S_i under declaration d_i . Since $d_i'(T_i) = d_i(T_i)$, $d_i'(S_i) \leq d_i(S_i)$, and ranking functions are monotone, this can occur only² if $S_i = T_i$. Thus $\mathcal{A}_i(d_i', \mathbf{d}_{-i}) = \mathcal{A}_i(d_i, \mathbf{d}_{-i})$ and hence $u_i(d_i, \mathbf{d}_{-i}) = u_i(d_i', \mathbf{d}_{-i})$ as claimed.

We conclude that $u_i(d_i', \mathbf{d}_{-i}) \geq u_i(d_i, \mathbf{d}_{-i})$ for all \mathbf{d}_{-i} . Suppose now that there exists some \mathbf{d}_{-i} and $S \subseteq M$ such that $d_i(S) > v_i(S)$ and $\mathcal{A}_i(d_i, \mathbf{d}_{-i}) = S$. In this case, our inequality is strict, since $u_i(d_i, \mathbf{d}_{-i}) = d_i(S) - v_i(S) < 0$. Thus, if this event occurs for any \mathbf{d}_{-i} , we conclude that b_i is strictly dominated by \underline{b}_i . \square

We conclude that the only way in which an overbidding strategy b_i is not strictly

²It is here that we make use of the fact that ties in rank are broken according to some arbitrary but fixed rule.

dominated by an easy-to-find alternative is if it only suggests overbidding on sets that can never be allocated to agent i . Since such sets are essentially irrelevant to the strategy of agent i , we can assume from this point onward that agents avoid the dominated strategy of overbidding.

5.3.1 Warmup: Pure Nash Equilibria

We are now ready to bound the price of anarchy of $\mathcal{M}_1(\mathcal{A})$. As a warmup, we will begin with a result for pure Nash equilibria, rather than the fully general BNE case. Note that the bound we obtain for pure equilibria is tighter than the bound we will obtain for the general case in Theorem 5.3.4.

Theorem 5.3.2 *Suppose \mathcal{A} is a monotone greedy c -approximate allocation rule for a combinatorial allocation problem. Then the price of anarchy of $\mathcal{M}_1(\mathcal{A})$ is at most c .*

Proof: Fix type profile \mathbf{v} and suppose $\mathbf{d} = \mathbf{b}(\mathbf{v})$ forms a pure Nash equilibrium. Let \mathbf{y} be an optimal allocation for \mathbf{v} , and let $\mathbf{x}(\cdot)$ denote the allocation rule for \mathcal{A} . Then recall that

$$\sum_i d_i(x_i(\mathbf{d})) \geq \frac{1}{c} \sum_i \theta_i(y_i, \mathbf{d}_{-i}) \quad (5.1)$$

by Lemma 5.2.3.

Choose arbitrarily small $\epsilon > 0$ and let d_i' be the single-minded declaration for set y_i at value $\theta_i(y_i, \mathbf{d}_{-i}) + \epsilon$. Then $x_i(d_i', \mathbf{d}_{-i}) = y_i$ (from the definition of critical prices) and hence $u_i(d_i', \mathbf{d}_{-i}) = v_i(y_i) - \theta_i(y_i, \mathbf{d}_{-i}) - \epsilon$. Since \mathbf{d} is a Nash equilibrium, it must be that

$$\begin{aligned} v_i(y_i) - \theta_i(y_i, \mathbf{d}_{-i}) - \epsilon &= u_i(d_i', \mathbf{d}_{-i}) \\ &\leq u_i(d_i, \mathbf{d}_{-i}) \\ &= v_i(x_i(\mathbf{d})) - d_i(x_i(\mathbf{d})). \end{aligned}$$

Summing over all i and applying (5.1) and Lemma 5.3.1 we have

$$\begin{aligned} \sum_i v_i(y_i) &\leq \sum_i \theta_i(y_i, \mathbf{d}_{-i}) - \sum_i d_i(x_i(\mathbf{d})) + \sum_i v_i(x_i(\mathbf{d})) + n\epsilon \\ &\leq (c-1) \sum_i d_i(x_i(\mathbf{d})) + \sum_i v_i(x_i(\mathbf{d})) + n\epsilon \\ &\leq c \sum_i v_i(x_i(\mathbf{d})) + n\epsilon \end{aligned}$$

which, taking $\epsilon \rightarrow 0$, implies

$$\begin{aligned} SW(\mathbf{x}(\mathbf{d}), \mathbf{v}) &= \sum_i v_i(x_i(\mathbf{d})) \\ &\geq \frac{1}{c} \sum_i v_i(y_i) \\ &= \frac{1}{c} SW_{OPT}(\mathbf{v}) \end{aligned}$$

as required. □

5.3.2 Existence of Pure Nash Equilibria

The power of Theorem 5.3.2 is marred by the fact that, for some problem instances, the mechanism $\mathcal{M}_1(\mathcal{A})$ is not guaranteed to have a pure Nash equilibrium. This is true even under the assumption that private valuations and payments are discretized, so that all values and payments are multiples of some arbitrarily small $\epsilon > 0$. A simple example is given below.

Example 5.3.3 *Consider an instance of the combinatorial auction problem with two objects, $M = \{a, b\}$, and three agents. Our feasibility constraint is that each agent can be assigned at most one object, and moreover agent 2 cannot be allocated object b and agent 3 cannot be allocated object a . Let \mathcal{A} be the greedy algorithm that ranks bids by value. Suppose the true types of the agents are as follows: $v_1(a) = 4$, $v_1(b) = 2$, $v_2(a) = 3$, $v_2(b) = 0$, $v_3(a) = 0$, and $v_3(b) = 3$.*

We now prove that no pure Nash equilibrium exists for this example, even if we assume that agents declare multiples³ of some $\epsilon > 0$. Assume for contradiction that there is a Nash equilibrium \mathbf{d} for type profile \mathbf{v} and mechanism $\mathcal{M}_1(\mathcal{A})$.

We know that agent 1 does not win item b with a payment greater than 2, as this would cause him negative utility (so he would certainly not be in equilibrium). Thus it must be that $\mathcal{A}_3(\mathbf{d}) = \{b\}$, since otherwise agent 3 could change his declaration to win $\{b\}$ and increase his utility. Thus, since agent 1 does not win item $\{b\}$, we conclude that $\mathcal{A}_1(\mathbf{d}) = \{a\}$, since otherwise agent 1 could change his declaration to win $\{a\}$ and increase his utility.

Now note that if $d_1(\{a\}) < 3$, agent 2 could increase his utility by making a winning declaration for $\{a\}$. Thus $d_1(\{a\}) \geq 3$, and hence $u_1(\mathbf{d}) \leq 4 - 3 = 1$. This also implies that $d_1(\{a\}) > d_1(\{b\})$, so agent 3 would win $\{b\}$ regardless of his bid. Thus, since agent 3 maximizes his utility up to an additive ϵ , it must be that $d_3(\{b\}) \leq \epsilon$. But then agent 1 could improve his utility by changing his declaration and bidding 0 for $\{a\}$ and 2ϵ for $\{b\}$, obtaining utility $2 - 2\epsilon > 1$. Therefore \mathbf{d} is not an equilibrium, a contradiction.

5.3.3 Bayes-Nash Equilibria

We are now ready to bound the mixed Bayesian price of anarchy for mechanism $\mathcal{M}_1(\mathcal{A})$.

Theorem 5.3.4 *Suppose \mathcal{A} is a monotone greedy c -approximate allocation rule for a combinatorial allocation problem. Then the Bayesian price of anarchy of $\mathcal{M}_1(\mathcal{A})$ is at most $c + O(c^2/e^c)$.*

Fix a product distribution \mathbf{F} over type profiles and let $\mathbf{b}(\cdot)$ be a (possibly mixed) Bayes-Nash equilibrium with respect to \mathbf{F} . Choose some type declaration \mathbf{v} and let $\mathbf{y}^{\mathbf{v}}$ denote an optimal allocation for \mathbf{v} . Following the proof of Theorem 5.3.2, we would like

³That is, our lack of pure equilibrium is not due to the possibility of infinitesimal improvements. One can also interpret our example as demonstrating that there is no $(1 + \epsilon)$ -approximate pure Nash equilibrium for small $\epsilon > 0$.

to bound the expected value of $\theta_i(y_i^{\mathbf{v}}, \mathbf{d}_{-i})$ with respect to $v_i(y_i^{\mathbf{v}})$ and $u_i(\mathbf{b}(\mathbf{v}))$ for each i . We encapsulate this bound in Lemma 5.3.6 and Corollary 5.3.7, below. This will allow us to use Lemma 5.2.3 to obtain a relation between the expected welfare of \mathcal{A} and the expected optimal welfare; this relationship is given in Lemma 5.3.5.

Lemma 5.3.5 *Suppose that \mathcal{A} is a c -approximate greedy algorithm and that there exist constants $\gamma \geq 0$ and $\sigma_i \in [0, c]$ for $i \in [n]$ such that, whenever \mathbf{b} is a Bayes-Nash equilibrium for $\mathcal{M}_1(\mathcal{A})$, it is the case that for all i , all v_i , and all $S \subseteq M$,*

$$\mathbf{E}_{\mathbf{v}_{-i}}[\theta_i(S, \mathbf{b}_{-i}(\mathbf{v}_{-i}))] \geq \gamma v_i(S) - \sigma_i \mathbf{E}_{\mathbf{v}_{-i}}[u_i(\mathbf{b}(\mathbf{v}))].$$

Then $\mathbf{E}_{\mathbf{v}}[SW(\mathcal{A}(\mathbf{b}(\mathbf{v})), \mathbf{v})] \geq \frac{\gamma}{c} \mathbf{E}_{\mathbf{v}}[SW_{OPT}(\mathbf{v})]$.

Lemma 5.3.6 *Suppose that \mathbf{b} is a Bayes-Nash equilibrium for mechanism $\mathcal{M}_1(\mathcal{A})$ and distribution \mathbf{F} . Then for all i , all v_i , and all $S \subseteq M$,*

$$\mathbf{E}_{\mathbf{v}_{-i}}[\theta_i(S, \mathbf{b}_{-i}(\mathbf{v}_{-i}))] \geq v_i(S) - \left(1 + \log \frac{v_i(S)}{\mathbf{E}_{\mathbf{v}_{-i}}[u_i(\mathbf{b}(\mathbf{v}))]}\right) \mathbf{E}_{\mathbf{v}_{-i}}[u_i(\mathbf{b}(\mathbf{v}))].$$

Before proving Lemmas 5.3.5 and 5.3.6, let us first show how they imply Theorem 5.3.4. We first note the following simple corollary of Lemma 5.3.6.

Corollary 5.3.7 *Suppose that \mathbf{b} is a Bayes-Nash equilibrium for mechanism $\mathcal{M}_1(\mathcal{A})$ and distribution \mathbf{F} . Then for all i , all v_i , and all $S \subseteq M$,*

$$\mathbf{E}_{\mathbf{v}_{-i}}[\theta_i(S, \mathbf{b}_{-i}(\mathbf{v}_{-i}))] \geq \gamma v_i(S) - \sigma_i \mathbf{E}_{\mathbf{v}_{-i}}[u_i(\mathbf{b}(\mathbf{v}))]$$

where $\gamma = \left(1 - \frac{c}{e^{c-1}}\right)$ and $\sigma_i = \min \left\{c, 1 + \log \frac{v_i(S)}{\mathbf{E}_{\mathbf{v}_{-i}}[u_i(\mathbf{b}(\mathbf{v}))]}\right\}$.

Proof: Fix agent i . By Lemma 5.3.6, we know

$$\mathbf{E}_{\mathbf{v}_{-i}}[\theta_i(S, \mathbf{b}_{-i}(\mathbf{v}_{-i}))] \geq v_i(S) - \left(1 + \log \frac{v_i(S)}{\mathbf{E}_{\mathbf{v}_{-i}}[u_i(\mathbf{b}(\mathbf{v}))]}\right) \mathbf{E}_{\mathbf{v}_{-i}}[u_i(\mathbf{b}(\mathbf{v}))]. \quad (5.2)$$

Note that if $\left(1 + \log \frac{v_i(S)}{\mathbf{E}_{\mathbf{v}_{-i}}[u_i(\mathbf{b}(\mathbf{v}))]}\right) \leq c$ then (5.2) immediately implies the desired result. We can therefore assume that $\left(1 + \log \frac{v_i(S)}{\mathbf{E}_{\mathbf{v}_{-i}}[u_i(\mathbf{b}(\mathbf{v}))]}\right) > c$, which implies

$$\frac{v_i(S)}{\mathbf{E}_{\mathbf{v}_{-i}}[u_i(\mathbf{b}(\mathbf{v}))]} > e^{c-1}.$$

Since the function $\frac{\log z}{z}$ is decreasing for $z > 1$, we conclude that

$$\left(1 + \log \frac{v_i(S)}{\mathbf{E}_{\mathbf{v}_{-i}}[u_i(\mathbf{b}(\mathbf{v}))]}\right) \mathbf{E}_{\mathbf{v}_{-i}}[u_i(\mathbf{b}(\mathbf{v}))] < c \cdot \frac{v_i(S)}{e^{c-1}}$$

and hence, by (5.2),

$$\mathbf{E}_{\mathbf{v}_{-i}}[\theta_i(S, \mathbf{b}_{-i}(\mathbf{v}_{-i}))] \geq v_i(S) - \frac{c}{e^{c-1}} v_i(S) = \left(1 - \frac{c}{e^{c-1}}\right) v_i(S)$$

which implies the desired result. \square

The proof of Theorem 5.3.4 now follows directly from Corollary 5.3.7 and Lemma 5.3.5.

Proof of Theorem 5.3.4 : Applying the bound from Corollary 5.3.7 to Lemma 5.3.5, we conclude that

$$\mathbf{E}_{\mathbf{v}}[SW(\mathcal{A}(\mathbf{b}(\mathbf{v})), \mathbf{v})] \geq \frac{1 - \frac{c}{e^{c-1}}}{c} \mathbf{E}_{\mathbf{v}}[SW_{OPT}(\mathbf{v})] \geq \frac{1}{c + \frac{2c^2}{e^{c-1}}} \mathbf{E}_{\mathbf{v}}[SW_{OPT}(\mathbf{v})]$$

as required. \square

We now complete the proof of Theorem 5.3.4 by proving Lemmas 5.3.5 and 5.3.6.

Proof of Lemma 5.3.5 : Fix distribution \mathbf{F} over type profiles and let $\mathbf{b}(\cdot)$ be a (possibly mixed) Bayes-Nash equilibrium with respect to \mathbf{F} . Choose some type declaration \mathbf{v} and let $\mathbf{y}^{\mathbf{v}}$ denote an optimal allocation for \mathbf{v} .

We know that for all $i \in [n]$ and \mathbf{v} ,

$$\mathbf{E}_{\mathbf{v}'_{-i}}[\theta_i(y_i^{\mathbf{v}}, \mathbf{b}_{-i}(\mathbf{v}'_{-i}))] \geq \gamma v_i(y_i^{\mathbf{v}}) - \sigma_i \mathbf{E}_{\mathbf{v}'_{-i}}[u_i(b_i(v_i), \mathbf{b}_{-i}(\mathbf{v}'_{-i}))].$$

Note the distinction between \mathbf{v}'_{-i} , over which we are taking expectations, and \mathbf{v}_{-i} , which is the type profile fixed to define $y_i^{\mathbf{v}}$. Now, summing over i and taking expectation over

all choices of \mathbf{v} , we have

$$\begin{aligned} & \mathbf{E}_{\mathbf{v}} \left[\sum_i \mathbf{E}_{\mathbf{v}'_{-i}} [\theta_i(y_i^{\mathbf{v}}, \mathbf{b}_{-i}(\mathbf{v}'_{-i}))] \right] \\ & \geq \gamma \mathbf{E}_{\mathbf{v}} \left[\sum_i v_i(y_i^{\mathbf{v}}) \right] \\ & \quad - \mathbf{E}_{\mathbf{v}} \left[\sum_i \sigma_i \mathbf{E}_{\mathbf{v}'_{-i}} [u_i(b_i(v_i), \mathbf{b}_{-i}(\mathbf{v}'_{-i}))] \right]. \end{aligned} \quad (5.3)$$

We now consider each of the three terms in (5.3). First, note that

$$\mathbf{E}_{\mathbf{v}} \left[\sum_i v_i(y_i^{\mathbf{v}}) \right] = \mathbf{E}_{\mathbf{v}} [SW_{OPT}(\mathbf{v})]. \quad (5.4)$$

Additionally,

$$\begin{aligned} & \mathbf{E}_{\mathbf{v}} \left[\sum_i \sigma_i \mathbf{E}_{\mathbf{v}'_{-i}} [u_i(b_i(v_i), \mathbf{b}_{-i}(\mathbf{v}'_{-i}))] \right] \\ & = \sum_i \sigma_i \mathbf{E}_{\mathbf{v}, \mathbf{v}'_{-i}} [u_i(b_i(v_i), \mathbf{b}_{-i}(\mathbf{v}'_{-i}))] \\ & = \mathbf{E}_{\mathbf{v}} \left[\sum_i \sigma_i u_i(\mathbf{b}(\mathbf{v})) \right] \\ & = \mathbf{E}_{\mathbf{v}} \left[\sum_i \sigma_i v_i(x_i(\mathbf{b}(\mathbf{v}))) \right] - \mathbf{E}_{\mathbf{v}, \mathbf{d}=\mathbf{b}(\mathbf{v})} \left[\sum_i \sigma_i d_i(x_i(\mathbf{b}(\mathbf{v}))) \right] \end{aligned} \quad (5.5)$$

where the final equality follows from the fact that our mechanism employs a first price payment scheme. Finally,

$$\begin{aligned} & \mathbf{E}_{\mathbf{v}} \left[\sum_i \mathbf{E}_{\mathbf{v}'_{-i}} [\theta_i(y_i^{\mathbf{v}}, \mathbf{b}_{-i}(\mathbf{v}'_{-i}))] \right] \\ & = \mathbf{E}_{\mathbf{v}, \mathbf{v}'} \left[\sum_i \theta_i(y_i^{\mathbf{v}}, \mathbf{b}_{-i}(\mathbf{v}'_{-i})) \right] \quad (\text{type independence}) \\ & \leq c \mathbf{E}_{\mathbf{v}, \mathbf{d}=\mathbf{b}(\mathbf{v}')} \left[\sum_i d'_i(x_i(\mathbf{d})) \right] \quad (\text{Lemma 5.2.3}) \\ & = c \mathbf{E}_{\mathbf{v}, \mathbf{d}=\mathbf{b}(\mathbf{v})} \left[\sum_i d_i(x_i(\mathbf{d})) \right]. \end{aligned} \quad (5.6)$$

Substituting (5.4), (5.5), and (5.6) into (5.3), we conclude that

$$\begin{aligned} c\mathbf{E}_{\mathbf{v}} \left[\sum_i d_i(x_i(\mathbf{d})) \right] &\geq \gamma\mathbf{E}_{\mathbf{v}}[SW_{OPT}(\mathbf{v})] - \mathbf{E}_{\mathbf{v}} \left[\sum_i \sigma_i v_i(x_i(\mathbf{b}(\mathbf{v}))) \right] \\ &\quad + \mathbf{E}_{\mathbf{v}, \mathbf{d}=\mathbf{b}(\mathbf{v})} \left[\sum_i \sigma_i d_i(x_i(\mathbf{b}(\mathbf{v}))) \right] \end{aligned}$$

and hence

$$\begin{aligned} \gamma\mathbf{E}_{\mathbf{v}}[SW_{OPT}(\mathbf{v})] &\leq \mathbf{E}_{\mathbf{v}} \left[\sum_i \sigma_i v_i(x_i(\mathbf{b}(\mathbf{v}))) \right] + \mathbf{E}_{\mathbf{v}, \mathbf{d}=\mathbf{b}(\mathbf{v})} \left[\sum_i (c - \sigma_i) d_i(x_i(\mathbf{d})) \right] \\ &\leq \mathbf{E}_{\mathbf{v}} \left[\sum_i \sigma_i v_i(x_i(\mathbf{b}(\mathbf{v}))) \right] + \mathbf{E}_{\mathbf{v}} \left[\sum_i (c - \sigma_i) v_i(x_i(\mathbf{b}(\mathbf{v}))) \right] \\ &= \mathbf{E}_{\mathbf{v}} \left[\sum_i c v_i(x_i(\mathbf{b}(\mathbf{v}))) \right] \\ &= c\mathbf{E}_{\mathbf{v}}[SW(\mathcal{A}(\mathbf{b}(\mathbf{v})), \mathbf{v})] \end{aligned}$$

where in the second inequality we used Lemma 5.3.1 plus the fact that $(c - \sigma_i) \geq 0$ for all i . Rearranging yields

$$\mathbf{E}_{\mathbf{v}}[SW(\mathcal{A}(\mathbf{b}(\mathbf{v})), \mathbf{v})] \geq \frac{\gamma}{c}\mathbf{E}_{\mathbf{v}}[SW_{OPT}(\mathbf{v})]$$

as required. \square

Proof of Lemma 5.3.6 : Fix a choice of i , v_i , and S . Since $\theta_i(S, \mathbf{d}_{-i}) \geq 0$ for all \mathbf{d}_{-i} , we have that

$$\begin{aligned} \mathbf{E}_{\mathbf{v}_{-i}}[\theta_i(S, \mathbf{b}_{-i}(\mathbf{v}_{-i}))] &\geq \int_0^{v_i(S)} \Pr[\theta_i(S, \mathbf{b}_{-i}(\mathbf{v}_{-i})) > z] dz \\ &= v_i(S) - \int_0^{v_i(S)} \Pr[\theta_i(S, \mathbf{b}_{-i}(\mathbf{v}_{-i})) \leq z] dz. \end{aligned}$$

Recall that $b_i(v_i)$ must maximize the expected utility of agent i . Choose any $z \geq 0$ and consider the alternative strategy d_i which places a single-minded bid of z on set S . Then since $b_i(v_i)$ is an optimal strategy, we have that

$$\begin{aligned} \mathbf{E}_{\mathbf{v}_{-i}}[u_i(\mathbf{b}(\mathbf{v}))] &\geq \mathbf{E}_{\mathbf{v}_{-i}}[u_i(d_i, \mathbf{b}_{-i}(\mathbf{v}_{-i}))] \\ &= (v_i(S) - z) \Pr[\theta_i(S, \mathbf{b}_{-i}(\mathbf{v}_{-i})) \leq z] \end{aligned}$$

for all $z \geq 0$. We conclude that $\Pr[\theta_i(S, \mathbf{b}_{-i}(\mathbf{v}_{-i})) \leq z] \leq \frac{\mathbf{E}_{\mathbf{v}_{-i}}[u_i(\mathbf{b}(\mathbf{v}))]}{(v_i(S) - z)}$ for all $0 \leq z < v_i(S)$. We also know that $\Pr[\theta_i(S, \mathbf{b}_{-i}(\mathbf{v}_{-i})) \leq z] \leq 1$ for all z . Write $r = v_i(S) - \mathbf{E}_{\mathbf{v}_{-i}}[u_i(\mathbf{b}(\mathbf{v}))]$. We then conclude that

$$\begin{aligned} \mathbf{E}_{\mathbf{v}_{-i}}[\theta_i(S, \mathbf{b}_{-i}(\mathbf{v}_{-i}))] &\geq v_i(S) - \int_0^r \frac{\mathbf{E}_{\mathbf{v}_{-i}}[u_i(\mathbf{b}(\mathbf{v}))]}{(v_i(S) - z)} dz - \int_r^{v_i(S)} 1 dz \\ &= v_i(S) - \mathbf{E}_{\mathbf{v}_{-i}}[u_i(\mathbf{b}(\mathbf{v}))] \int_{\mathbf{E}_{\mathbf{v}_{-i}}[u_i(\mathbf{b}(\mathbf{v}))]}^{v_i(S)} \frac{1}{y} dy - \mathbf{E}_{\mathbf{v}_{-i}}[u_i(\mathbf{b}(\mathbf{v}))] \\ &= v_i(S) - \left(1 + \log \frac{v_i(S)}{\mathbf{E}_{\mathbf{v}_{-i}}[u_i(\mathbf{b}(\mathbf{v}))]}\right) \mathbf{E}_{\mathbf{v}_{-i}}[u_i(\mathbf{b}(\mathbf{v}))] \end{aligned}$$

as required. \square

This completes the proof of Theorem 5.3.4. It may be tempting to conjecture that the (exponentially small) loss in approximation factor in Theorem 5.3.4 is simply an artifact of the analysis, and that the Bayesian price of anarchy of $\mathcal{M}_1(\mathcal{A})$ is actually c . However, we now show by way of an example that this loss is crucial: there exist instances in which the Bayesian price of anarchy is strictly greater than c .

Proposition 5.3.8 *For any $c \geq 2$, there is a combinatorial allocation problem \mathcal{P} and a non-adaptive greedy algorithm \mathcal{A} such that \mathcal{A} is a c -approximation for \mathcal{P} , and the mixed price of anarchy for $\mathcal{M}_1(\mathcal{A})$ is $c + \Omega(c^2/e^{4c})$.*

Proof: We begin by describing our combinatorial allocation problem. We choose a parameter $k > c$ that will be fixed later. Our auction has $ck + k$ objects, which we label a_{ij} for $i \in [k], j \in [c]$ and b_i for $i \in [k]$. There are $4k$ agents, labelled A_i, B_i, C_i , and D_i for $i \in [k]$. Our feasibility constraints are as follows. Each agent B_i or C_i can receive only set $\{a_{i1}\}$ or \emptyset . Each agent D_i can receive set $\{a_{i1}, a_{k1}\}$ or \emptyset . Each agent A_i can receive either set $\{a_{i1}, a_{i2}, \dots, a_{ic}\}$, set $\{b_i\}$, or \emptyset . Under these restrictions, an allocation is feasible if each object is assigned to at most one agent.

Let \mathcal{A} be the non-adaptive greedy algorithm that orders bids by density: i.e. with priority function $r(i, S, v) = v/|S|$ when S is a feasible set for agent i . We claim that when $c \geq 2$, this algorithm obtains a c -approximation for the above combinatorial auction. To

see this, note that the (unique) set that can be allocated to any agent B_i , C_i , or D_i intersects sets of size at most c times larger, so if the greedy algorithm allocates to one of these agents for a value of v , the total value of intersecting sets in the optimal solution is at most cv . On the other hand, if the greedy algorithm allocates $\{b_i\}$ to agent A_i , this conflicts only with the allocation of set $\{a_{i1}, \dots, a_{ic}\}$ to agent A_i , which again has value at most c times greater. Finally, suppose that the greedy algorithm allocates set $\{a_{i1}, \dots, a_{ic}\}$ to agent A_i , say with value vc (i.e. value density v). This allocation can conflict only with a single allocation to an agent B_i , C_i , or D_i plus an allocation of $\{b_i\}$ to agent A_i , which comprises a total of at most 3 objects. Since the greedy algorithm allocates by density, the total value of the conflicted bids is at most $3v$. Since $c \geq 2$, we conclude that the allocation of $\{a_{i1}, \dots, a_{ic}\}$ to agent A_i is within a factor of c of the value of any intersecting sets in the optimal allocation.

Consider now the following instance of this problem, specified by the following agent types.

- For $1 \leq i \leq k - 1$, agent A_i desires $\{a_{i1}, a_{i2}, \dots, a_{ic}\}$ for value $k - i$ and $\{b_i\}$ for value 0.
- Agent A_k desires $\{a_{k1}, a_{k2}, \dots, a_{kc}\}$ for value k and $\{b_k\}$ for value 1.
- For $1 \leq i \leq k$, agents B_i and C_i both desire set $\{a_{i1}\}$ for value $(k - i)/c$.
- For $1 \leq i \leq k$, agent D_i desires set $\{a_{i1}, a_{k1}\}$ for value $2(k - i)/c$.

Note that agent A_k has a value density of k/c for the desired set $\{a_{k1}, \dots, a_{kc}\}$, and each agent A_i with $i < k$ has value density $(k - i)/c$ for desired set $\{a_{i1}, \dots, a_{ic}\}$. Also, the agents B_i , C_i , and D_i have a value density of $(k - i)/c$ for their desired sets.

We can suppose that for any i , \mathcal{A} would break a tie between agents A_i , B_i , C_i , and/or D_i first in favour of D_i , then in favour of B_i , then A_i . We can also assume that \mathcal{A} breaks ties between multiple desired sets for agent A_i in favour of $\{b_i\}$. Finally, we can assume

that if agents declare the zero valuation, then \mathcal{A} will favour allocating non-empty sets over allocating the empty set.

We now describe a mixed Nash equilibrium for this problem instance. Each agent A_i declares the zero valuation. Each agent B_i and C_i declares his valuation truthfully. Each agent D_i will declare his valuation truthfully with some probability p_i , and will otherwise declare the zero valuation. We choose $p_i = \frac{1}{i+1}$.

What is the outcome when agents bid in this way? First, each agent A_i is allocated set $\{b_i\}$ (due to our assumed tie-breaking). For the items a_{ij} , only items with $j = 1$ will be allocated. For $i < k$, if agents D_1, \dots, D_{i-1} declare the zero allocation and D_i does not, then object a_{1i} will be allocated to D_i . If not, then item a_{1i} will be allocated to agent B_i . Item a_{k1} will be allocated to D_i where i is the smallest such that D_i does not declare the zero valuation, or B_k if D_1, \dots, D_k all declare the zero valuation.

We now argue that this distribution of declarations is indeed a mixed Nash equilibrium. With probability 1, no agent B_i , C_i , or D_i can obtain positive utility from any declaration (since their desired sets conflict with other bids of the same value density), so their distributions over declarations that obtain utility 0 are necessarily optimal. Furthermore, for each $i < k$, agent A_i cannot obtain positive utility so his bidding strategy is also optimal. Agent A_k obtains utility 1; his only hope for obtaining more utility is to declare a value less than $k - 1$ for set $\{a_{k1}, \dots, a_{kc}\}$. However, if he declares some value $k - z$ with $z > 1$, say with $x = \lceil z \rceil$, then he can win his desired set only if bidders D_1, \dots, D_{x-1} all bid the zero valuation, since otherwise an agent D_j with $j < x$ would win his desired set, blocking the bid by agent A_k . The probability that bidders D_1, \dots, D_{x-1} all declare the zero valuation is $\frac{1}{2} \cdot \frac{2}{3} \cdots \frac{x-1}{x} = \frac{1}{x} \leq \frac{1}{z}$. Thus, for any z , agent A_k can obtain utility z with probability at most $1/z$, for an expected utility of at most 1. The given declaration by agent A_k is therefore optimal.

We will now bound the social efficiency of this equilibrium. The optimal obtainable welfare is $k + \sum_{i=1}^{k-1} (k - i) = \frac{1}{2}k(k + 1)$, by allocating set $\{a_{i1}, \dots, a_{ic}\}$ to agent A_i for

all i . In the equilibrium we've described, object b_k is allocated to agent A_k for a value of 1 and each object a_{i1} for $i < k$ is allocated to either B_i or D_i at a per-item value of $(k - i)/c$. For each $i < k$, object a_{1k} will be allocated to bidder D_i precisely if bidders D_1, \dots, D_{j-1} declare the zero valuation but D_i does not, which occurs with probability $\frac{1}{i(i+1)}$. Object a_{1k} will be allocated to either B_k or D_k with the remaining probability, which is $\frac{1}{k}$. Noting that each of B_i and D_i has a per-item value of $(k - i)/c$ for their desired sets, we conclude that the expected total value obtained is

$$\begin{aligned} & 1 + \sum_{i < k} \frac{k - i}{c} + \sum_{i < k} \frac{1}{i(i+1)} \cdot \frac{k - i}{c} + \frac{1}{k} \cdot \frac{k - k}{c} \\ &= 1 + \frac{1}{c} \left[\frac{1}{2}(k^2 - k) + k - \sum_{i < k} \frac{1}{i+1} - 1 \right] \\ &= 1 + \frac{1}{c} \left[\frac{1}{2}(k^2 + k) - H_k \right] \end{aligned}$$

where H_k is the k th harmonic number.

We conclude that the mixed price of anarchy for this mechanism is at least

$$\frac{\frac{1}{2}(k^2 + k)}{1 + \frac{1}{c} \left[\frac{1}{2}(k^2 + k) - H_k \right]} > c \left(\frac{k^2 + k}{k^2 + k + 2c - 2 \ln k} \right)$$

where we used the fact that $H_k > \ln k$. Choose $k = e^{2c}$. Then our mechanism has mixed price of anarchy at least

$$c \left(\frac{e^{4c} + e^{2c}}{e^{4c} + e^{2c} - 2c} \right) > c \left(\frac{e^{4c}}{e^{4c} - c} \right) > c \left(1 + \frac{c}{e^{4c}} \right)$$

as required. □

5.3.4 Correlated Types

Recall that our bound upon the Bayesian Price of Anarchy for our first-price mechanism required that agent types be distributed independently. In this Section we give an alternative bound that allows agent types to be arbitrarily correlated. The key to this analysis is in considering a deviating behaviour for each agent that does not depend on

the other agents' types. The particular deviation we will consider is that of bidding half of one's true value for every set. Our analysis will additionally require that our original greedy algorithm be non-adaptive.

Theorem 5.3.9 *Suppose \mathcal{A} is a c -approximate non-adaptive greedy algorithm for a combinatorial allocation problem. Then $\mathcal{M}_1(\mathcal{A})$ has Correlated Bayesian Price of Anarchy at most $4c$, for any type distribution \mathbf{F} .*

The key to this result lies in the following lemma, whose statement is motivated by the smoothness condition of [82].

Lemma 5.3.10 *Suppose \mathcal{A} is a c -approximate non-adaptive greedy algorithm for a combinatorial allocation problem. Then for all type profiles \mathbf{v} and all strategy profiles $\mathbf{b}(\cdot)$,*

$$\sum_i u_i(v_i/2, \mathbf{b}_{-i}(\mathbf{v}_{-i})) \geq \frac{1}{2c} SW_{OPT}(\mathbf{v}) - SW(\mathcal{A}(\mathbf{b}(\mathbf{v})), \mathbf{v}).$$

Proof: Let \mathbf{y} denote the optimal allocation for type profile \mathbf{v} . Choose agent i , and consider the outcome of \mathcal{A} on input profile $(v_i/2, \mathbf{b}_{-i}(\mathbf{v}_{-i}))$. Let $x_i = \mathcal{A}_i(v_i/2, \mathbf{b}_{-i}(\mathbf{v}_{-i}))$. Note that it must either be that $\theta_i(y_i, \mathbf{b}_{-i}(\mathbf{v}_{-i})) \geq \frac{1}{2}v_i(y_i)$ or not. In the latter case, agent i must obtain some allocation x_i with $r(i, x_i, v_i(x_i)/2) \geq r(i, y_i, v_i(y_i)/2)$. Since \mathcal{A} is a non-adaptive greedy algorithm, this then implies that $v_i(x_i) \geq \frac{1}{c}v_i(y_i)$, since otherwise \mathcal{A} would obtain less than a $\frac{1}{c}$ fraction of the optimal social welfare on the input in which agent i places bids only on sets x_i and y_i , and all other agents bid 0.

We conclude that for all i , either $\theta_i(y_i, \mathbf{b}_{-i}(\mathbf{v}_{-i})) > \frac{1}{2}v_i(y_i)$ or else $v_i(x_i) \geq \frac{1}{c}v_i(y_i)$. Let $N = \{i \mid \theta_i(y_i, \mathbf{b}_{-i}(\mathbf{v}_{-i})) > \frac{1}{2}v_i(y_i)\}$ be the set of agents for which the former condition holds. We then note that

$$\sum_{i \in N} \frac{1}{2}v_i(y_i) < \sum_{i \in N} \theta_i(y_i, \mathbf{b}_{-i}(\mathbf{v}_{-i})) \leq c SW(\mathcal{A}(\mathbf{b}(\mathbf{v})), \mathbf{v})$$

where the second inequality is due to Lemma 5.2.3. Furthermore, since $v_i(x_i) \geq \frac{1}{c}v_i(y_i)$ for all $i \notin N$, we have

$$\sum_{i \notin N} \frac{1}{2}v_i(y_i) \leq \sum_{i \notin N} \frac{c}{2}v_i(x_i(v_i/2, \mathbf{b}_{-i}(\mathbf{v}_{-i}))) \leq c \sum_i u_i(v_i/2, \mathbf{b}_{-i}(\mathbf{v}_{-i}))$$

where the second inequality follows because we are using the first-price payment scheme. Combining these inequalities yields

$$\sum_i u_i(v_i/2, \mathbf{b}_{-i}(\mathbf{v}_{-i})) + SW(\mathcal{A}(\mathbf{b}(\mathbf{v})), \mathbf{v}) \geq \frac{1}{2c} SW_{OPT}(\mathbf{v})$$

as required. \square

Theorem 5.3.9 now follows easily from Lemma 5.3.10. Recall that Lemma 5.3.10 holds for all strategy profiles, not just strategies in equilibrium. If we take \mathbf{b} to be an equilibrium profile under type distribution \mathbf{F} , then

$$\begin{aligned} E_{\mathbf{v}}[SW(\mathcal{A}(\mathbf{b}(\mathbf{v})), \mathbf{v})] &\geq E_{\mathbf{v}} \left[\sum_i u_i(\mathbf{b}(\mathbf{v})) \right] \\ &= \sum_i E_{v_i} E_{\mathbf{v}_{-i}|v_i} [u_i(b_i(v_i), \mathbf{b}_{-i}(\mathbf{v}_{-i}))] \\ &\geq \sum_i E_{v_i} E_{\mathbf{v}_{-i}|v_i} \left[u_i \left(\frac{v_i}{2}, \mathbf{b}_{-i}(\mathbf{v}_{-i}) \right) \right] \\ &= E_{\mathbf{v}} \left[\sum_i u_i \left(\frac{v_i}{2}, \mathbf{b}_{-i}(\mathbf{v}_{-i}) \right) \right] \\ &\geq E_{\mathbf{v}} \left[\frac{1}{2c} \sum_i SW_{OPT}(\mathbf{v}) - SW(\mathcal{A}(\mathbf{b}(\mathbf{v})), \mathbf{v}) \right] \quad (\text{Lemma 5.3.10}) \end{aligned}$$

from which we conclude that

$$E_{\mathbf{v}}[SW(\mathcal{A}(\mathbf{b}(\mathbf{v})), \mathbf{v})] \geq \frac{1}{4c} E_{\mathbf{v}} \left[\sum_i OPT(\mathbf{v}) \right]$$

completing the proof of Theorem 5.3.9.

5.4 Critical-Price Mechanisms

We have shown that the price of anarchy for $\mathcal{M}_1(\mathcal{A})$ is $c + O(c^2/e^c)$ when \mathcal{A} is a c -approximation. The error term in this bound vanishes as c grows, but can be significantly when c is small (recall from the proof of Theorem 5.3.4 that the constant hidden by the asymptotic notation is $2e$). Can we do away with this loss in approximation factor by modifying our mechanism? Intuitively speaking, the extra error term is introduced

because an agent may not know how to bid in order to obtain some set S and pay the minimum possible amount for it. This uncertainty is inherent in the first-price payment scheme. An alternative, the critical-price payment rule, does not exhibit this problem: under critical pricing, an agent that wins a set always pays the optimal price for it. We now study mechanisms that apply this alternative payment rule. We will find that the price of anarchy for the critical price mechanism is actually greater than that of the first-price mechanism in general for large c , but can be better for small c . Also, the price of anarchy for the critical price mechanism can be improved to c , removing the error term entirely, in certain special cases.

5.4.1 Calculating Critical Prices

We will first discuss the calculation of critical prices. For many allocation algorithms (such as all of the algorithms discussed in Section 5.7, the calculation of critical prices is a simple task, which can be performed in parallel with the computation of an allocation profile. We leave the development of such pricing methods to the creators of the allocation algorithms to which our reduction may be applied. However, even if a specially-tailored algorithm for computing exact critical prices is not available, we note that critical prices for a given black-box greedy algorithm can be determined to within an additive ϵ error in polynomial time via simple binary search. Thus, assuming that valuation space is discretized by multiples of ϵ , critical prices can be determined efficiently. If valuation space is continuous, then our interpretation is that any equilibrium for the (exact) critical-price mechanism will be an (additive) ϵ -approximate equilibrium for a mechanism that uses ϵ -approximate critical prices.

We now describe the procedure for determining critical prices in more detail. Fix greedy allocation rule \mathcal{A} , agent i , and declarations \mathbf{d} . Suppose that $\mathcal{A}_i(d_i, \mathbf{d}_{-i}) = S$. We wish to resolve the value of $\theta_i(S, \mathbf{d}_{-i})$ in the range $[0, d_i(S)]$ using binary search in the following way. For all $z \geq 0$, write d_i^z for the single-minded declaration for set

S at value z . Given query value $z \in [0, d_i(S)]$, we check if $\mathcal{A}_i(d_i^z, d_i) = S$. If so, decrease the value of z ; otherwise, increase the value of z . Since \mathcal{A} is monotone, we have that $\mathcal{A}_i(d_i^z, \mathbf{d}_{-i}) = S$ if and only if $z > \theta_i(S, \mathbf{d}_{-i})$. This procedure resolves the value of v to within ϵ in $O(\log d_i(S)/\epsilon)$ iterations. Thus, for any given input to mechanism $\mathcal{M}_{crit}(\mathcal{A})$, the critical prices for all agents' allocated sets can be found in $O(n \log(v_{max}/\epsilon))$ invocations of algorithm \mathcal{A} , where $v_{max} = \max_{i,S} d_i(S)$.

5.4.2 Bayes-Nash Equilibria

We now wish to analyze the Bayesian price of anarchy for critical price mechanisms. Unfortunately, Lemma 5.3.1, which plays a crucial part in the proof of Theorem 5.3.4, fails to hold for the critical payment rule: there is no guarantee that rational agents will not overbid for mechanism $\mathcal{M}_{crit}(\mathcal{A})$. Indeed, there are settings in which an agent may be *strictly* better off by overbidding, even in full-information settings, as the following example shows.

Example 5.4.1 *Consider a combinatorial auction with 3 objects, $\{a, b, c\}$, and 3 bidders, under the feasibility restriction that each agent can be allocated at most one object. Let \mathcal{A} be the greedy algorithm that orders bids by value. Suppose the types of the players are as follows: $t_1(b) = 2$, $t_1(c) = 4$, $t_2(c) = 3$, $t_3(a) = 1$, $t_3(b) = 6$, and all other values are 0. Consider the following bidding strategies for agents 2 and 3: bidder 2 declares truthfully with probability 1, and bidder 3 either declares single-mindedly for a with value 1, or single-mindedly for b with value 6, each with equal probability.*

How should agent 1 declare to maximize utility? We can limit our analysis to pure strategies (as any optimal randomized strategy has only optimal strategies in its support). Suppose agent 1 does not overbid and declares at most 2 for object b . If he also declares at least 3 for object c , then he wins c with probability 1 for an expected utility of 1. If he doesn't declare at least 3 for object c , then he wins b with probability 1/2 and nothing otherwise, again for an expected utility of 1. So agent 1 can gain a utility of at most 1

if he does not overbid. If, however, he declares 5 for b and 4 for c , then he wins b with probability $1/2$ and wins c otherwise, for an expected utility of $3/2$. If agent 1 bids in this way, the resulting combination of strategies forms a mixed Nash equilibrium. Thus, in mixed equilibria, an agent may strictly improve his utility by overbidding.

We get around the issue of overbidding by directly assuming that agents do not overbid. Such an assumption is most reasonable in settings where agents can be presumed to be averse to risking negative utility. Given that agents will not overbid, a simple modification of Theorem 5.3.4 yields a sharpened result. Furthermore, our result degrades gracefully if agents bid only at approximate equilibria.

Theorem 5.4.2 *Suppose \mathcal{A} is a c -approximate greedy allocation rule, and that $\mathbf{b}(\cdot)$ is a $(1 + \gamma)$ -approximate Bayes-Nash equilibrium of $\mathcal{M}_{crit}(\mathcal{A})$ in which agents do not overbid. Then the expected welfare when agents declare according to \mathbf{b} is a $(c+1+\gamma)$ approximation to the expected optimal welfare.*

The proof follows in the same manner as Theorem 5.3.4, applying the following variant of Lemma 5.3.6.

Lemma 5.4.3 *Suppose that \mathbf{b} is a $(1+\gamma)$ -approximate Bayes-Nash equilibrium for mechanism $\mathcal{M}_{crit}(\mathcal{A})$ and distribution \mathbf{F} . Then for all i , all v_i , and all $S \subseteq M$,*

$$\mathbf{E}_{\mathbf{v}_{-i}}[\theta_i(S, \mathbf{b}_{-i}(\mathbf{v}_{-i}))] \geq v_i(S) - (1 + \gamma)\mathbf{E}_{\mathbf{v}_{-i}}[v_i(x_i(b_i(v_i), \mathbf{b}_{-i}(\mathbf{v}_{-i})))]$$

Proof: Choose any i , v_i , and S . Let d_i be a single-minded declaration for set S at value $v_i(S)$, and consider a strategy under which agent i declares d_i when his type is v_i . Under this strategy, the expected utility of agent i with type v_i is

$$\begin{aligned} \mathbf{E}_{\mathbf{v}_{-i}}[u_i(d_i, \mathbf{b}_{-i}(\mathbf{v}_{-i}))] &\geq \mathbf{E}_{\mathbf{v}_{-i}}[\max\{v_i(S) - \theta_i(S, \mathbf{b}_{-i}(\mathbf{v}_{-i})), 0\}] \\ &\geq v_i(S) - \mathbf{E}_{\mathbf{v}_{-i}}[\theta_i(S, \mathbf{b}_{-i}(\mathbf{v}_{-i}))]. \end{aligned} \tag{5.7}$$

Since b_i is a $(1 + \gamma)$ -approximate equilibrium strategy for agent i , it must be that

$$\begin{aligned} \mathbf{E}_{\mathbf{v}_{-i}}[u_i(d_i, \mathbf{b}_{-i}(\mathbf{v}_{-i}))] &\leq (1 + \gamma)\mathbf{E}_{\mathbf{v}_{-i}}[u_i(b_i(v_i), \mathbf{b}_{-i}(\mathbf{v}_{-i}))] \\ &\leq (1 + \gamma)\mathbf{E}_{\mathbf{v}_{-i}}[v_i(x_i(b_i(v_i), \mathbf{b}_{-i}(\mathbf{v}_{-i})))]. \end{aligned} \quad (5.8)$$

Combining equations (5.7) and (5.8) leads to the desired result. \square

Following the proof of Theorem 5.3.4, we conclude that for all equilibria \mathbf{b} , if we write $\mathbf{y}^{\mathbf{v}}$ for an optimal allocation for any given type profile \mathbf{v} , then

$$\begin{aligned} &\mathbf{E}_{\mathbf{v}} \left[\sum_i \mathbf{E}_{\mathbf{v}'_{-i}}[\theta_i(y_i^{\mathbf{v}}, \mathbf{b}_{-i}(\mathbf{v}_{-i}))] \right] \\ &\geq \mathbf{E}_{\mathbf{v}} \left[\sum_i v_i(y_i^{\mathbf{v}}) \right] \\ &\quad - (1 + \gamma)\mathbf{E}_{\mathbf{v}} \left[\sum_i \mathbf{E}_{\mathbf{v}'_{-i}}[v_i(x_i(b_i(v_i), \mathbf{b}_{-i}(\mathbf{v}_{-i})))] \right]. \end{aligned} \quad (5.9)$$

Just as in the proof of Theorem 5.3.4, we obtain the bounds

$$\begin{aligned} \mathbf{E}_{\mathbf{v}} \left[\sum_i v_i(y_i^{\mathbf{v}}) \right] &= \mathbf{E}_{\mathbf{v}}[SW_{OPT}(\mathbf{v})], \\ \mathbf{E}_{\mathbf{v}} \left[\sum_i \mathbf{E}_{\mathbf{v}'_{-i}}[v_i(x_i(b_i(v_i), \mathbf{b}_{-i}(\mathbf{v}_{-i})))] \right] &= \mathbf{E}_{\mathbf{v}}[SW(\mathcal{A}(\mathbf{b}(\mathbf{v})), \mathbf{v})], \\ \mathbf{E}_{\mathbf{v}} \left[\sum_i \mathbf{E}_{\mathbf{v}'_{-i}}[\theta_i(y_i^{\mathbf{v}}, \mathbf{b}_{-i}(\mathbf{v}_{-i}))] \right] &\leq c\mathbf{E}_{\mathbf{v}}[SW(\mathcal{A}(\mathbf{b}(\mathbf{v})), \mathbf{v})] \end{aligned}$$

which, taken together with (5.9), completes the proof of Theorem 5.4.2. Note that for the third bound above, we use the assumption that agents do not overbid in place of Lemma 5.3.1. \square

For the special case that $\mathbf{b}(\cdot)$ is an exact Bayes-Nash equilibrium, we obtain a Bayesian Price of Anarchy of $(c + 1)$. We next show that this gap between the approximation factor of the original algorithm and the price of anarchy of the critical price mechanism is essential for large c . For any $c \geq 1$ we exhibit a combinatorial allocation problem and a non-adaptive greedy algorithm \mathcal{A} such that the approximation factor of \mathcal{A} is $c + \frac{1}{c}$ but the (pure) price of anarchy of $\mathcal{M}_{crit}(\mathcal{A})$ is $c + 1$.

Proposition 5.4.4 *For any $c \geq 1$, there is a combinatorial allocation problem \mathcal{P} and a non-adaptive greedy algorithm \mathcal{A} such that \mathcal{A} is a $(c + \frac{1}{c})$ -approximation for \mathcal{P} , and the pure price of anarchy for $\mathcal{M}_{crit}(\mathcal{A})$ (assuming agents do not overbid) is $c + 1$.*

Proof: Consider a combinatorial auction problem with two objects a, b and two players, under the restriction that each player can be allocated at most one object and player 2 cannot be allocated object b . Algorithm \mathcal{A} will be the following non-adaptive greedy algorithm: if $v_1(a) \geq \frac{1}{c}v_2(a)$ and $v_1(a) \geq cv_1(b)$ then allocate a to player 1 and \emptyset to player 2; otherwise allocate b to player 1 and a to player 2. Note that this is a $c + \frac{1}{c}$ approximation algorithm, since whenever the algorithm allocates a to player 1 we have $v_2(a) + v_1(b) \leq (c + \frac{1}{c})v_1(a)$, and whenever the algorithm allocates a to player 2 we have $v_1(a) \leq c(v_1(b) + v_2(b))$.

Consider the mechanism $\mathcal{M}_{crit}(\mathcal{A})$, and suppose that the agents have a type profile in which $v_1(a) = v_1(b) = 1$ and $v_2(a) = c$. Then the declaration profile $d_1(a) = 1, d_1(b) = 0$, and $d_2(a) = 0$ is in equilibrium, since agent 1 cannot improve upon his utility of 1 and agent 2 cannot affect the outcome without paying at least $\theta_2(a, d_1) = c$, for a utility of 0. The social welfare at this equilibrium is 1, but a total of $c + 1$ is possible by allocating a to player 2 and allocating b to player 1. Thus the price of anarchy for $\mathcal{M}_{crit}(\mathcal{A})$ is at least $c + 1$. \square

We conclude that, in general, the bound in Theorem 5.4.2 cannot be improved beyond $c + 1 - \theta(\frac{1}{c})$. However, we note that the bound can be improved to c in certain important special cases. This follows directly from a corresponding improvement to Lemma 5.2.3, discussed in Section 5.6.

5.5 Combining Mechanisms

A standard technique in the design of allocation rules is to consider both a greedy rule that favours allocation of small sets, and a simple rule that allocates all objects to a

single bidder, and apply whichever solution obtains the better result [11, 18, 71]. When bidders are single-minded, such a combination rule will be incentive-compatible [71]. We would like to extend our results to cover rules of this form, but the price of anarchy for such a rule (with either the first-price or critical-price payment scheme) may be much worse than its combinatorial approximation ratio. Consider the following example.

Example 5.5.1 *Consider the combinatorial auction problem. Suppose \mathcal{A} is the non-adaptive greedy algorithm with priority rule $r(i, S, v) = v$ if $|S| \leq \sqrt{m}$, and $r(i, S, v) = 0$ otherwise. Let \mathcal{A}' be the non-adaptive greedy algorithm with priority rule $r(i, S, v) = v$ if $S = M$, and $r(i, S, v) = 0$ otherwise. Then \mathcal{A}' simply allocates the set of all objects to the player that declares the highest value for it. Let \mathcal{A}_{max} be the allocation rule that applies whichever of \mathcal{A} or \mathcal{A}' obtains the better result; that is, on input \mathbf{d} , \mathcal{A}_{max} returns $\mathcal{A}(\mathbf{d})$ if $SW(\mathcal{A}(\mathbf{d}), \mathbf{d}) > SW(\mathcal{A}'(\mathbf{d}), \mathbf{d})$, otherwise returns $\mathcal{A}'(\mathbf{d})$. It is known that \mathcal{A}_{max} is a $O(\sqrt{m})$ approximate algorithm [71].*

Our instance of the CA problem is the following. We have $n = m \geq 2$, say with $M = \{a_1, \dots, a_m\}$. Choose $\epsilon > 0$ arbitrarily small. For each i , the private type of agent i , v_i , is the pointwise maximum of two single-minded valuation functions: one for set $\{a_i\}$ at value 1, and the other for set M at value $1 + \epsilon$. An optimal allocation profile for \mathbf{v} would assign $\{a_i\}$ to each agent i , for a total welfare of m .

We construct a declaration profile as follows. For each i , d_i is the single-minded valuation function for set M at value $1 + \epsilon$. On input \mathbf{d} , \mathcal{A}_{max} will assign M to some agent, for a total welfare of $1 + \epsilon$. Also, \mathbf{d} is a pure Nash equilibrium for $\mathcal{M}_1(\mathcal{A}_{max})$ and $\mathcal{M}_{crit}(\mathcal{A}_{max})$: all agents receive a utility of 0, and there is no way for any single agent to obtain positive utility by deviating from \mathbf{d} . Taking $\epsilon \rightarrow 0$, we conclude that the price of anarchy for any of these mechanisms is $\Omega(m)$, which does not match the combinatorial $O(\sqrt{m})$ approximation ratio of \mathcal{A}_{max} .

In light of the above example, we consider a different way to combine two rules: we implement each rule as a separate mechanism, then randomly choose between the two

mechanisms with equal probability. For many examples of interest (eg. combinatorial auctions, see Section 5.7) the resulting randomized allocation rule obtains (in expectation) the same worst-case combinatorial approximation ratio as applying the better of the two rules for each input. Moreover, the price of anarchy results of this paper can be made to carry over to such randomized mechanisms, as we now formalize.

Let \mathcal{A} be any greedy allocation rule that never allocates M to any agent, and let \mathcal{A}' be the allocation rule that allocates M to the agent i that maximizes $d_i(M)$. The restriction on \mathcal{A} is motivated by our intuition that \mathcal{A} favours allocations of small sets; it is without loss of generality for many algorithms of interest (eg. combinatorial auctions, again see Section 5.7). We write $\mathcal{M}_1(\mathcal{A}, \mathcal{A}')$ for the mechanism that flips a fair coin, and if it lands heads it executes $\mathcal{M}_1(\mathcal{A})$, otherwise executes $\mathcal{M}_1(\mathcal{A}')$. For this mechanisms, we will *allow input valuations to be non-monotone* with respect to set M ; that is, we allow declarations in which $d_i(M) < d_i(S)$ for $S \subseteq M$. Note then that our mechanism is not technically a direct revelation mechanism, as an agent's input is not necessarily a monotone valuation function.

Theorem 5.5.2 *Suppose that $\mathcal{A}, \mathcal{A}'$ are as described above and $SW(\mathcal{A}, \mathbf{d}) + SW(\mathcal{A}', \mathbf{d}) \geq \frac{1}{c} SW_{opt}(\mathbf{d})$ for every declaration profile \mathbf{d} . Then $\mathcal{M}_1(\mathcal{A}, \mathcal{A}')$ obtains a $2(c + O(c^2/e^c))$ approximation at every mixed BNE.*

Proof: Since the portions of agent declarations relevant to $\mathcal{M}_1(\mathcal{A})$ and $\mathcal{M}_1(\mathcal{A}')$ are independent, an agent will optimize his declaration for $\mathcal{M}_1(\mathcal{A}, \mathcal{A}')$ by optimizing for $\mathcal{M}_1(\mathcal{A})$ and $\mathcal{M}_1(\mathcal{A}')$ separately. Theorem 5.3.4 then immediately implies the desired result, as an equilibrium for $\mathcal{M}_1(\mathcal{A}, \mathcal{A}')$ must be a combination of an equilibrium for $\mathcal{M}_1(\mathcal{A})$ and an equilibrium for $\mathcal{M}_1(\mathcal{A}')$. \square

5.6 Tightening Results for Special Cases

In this section we show how to tighten the results of Lemma 5.2.3 for certain special cases of allocation problems and greedy algorithms. This allows us to obtain a sharper bound in Theorem 5.4.2. We say that a combinatorial allocation problem is *player symmetric* if the feasibility constraints do not depend on the labels of the players, and *object symmetric* if they do not depend on the labels of the objects. We say that a greedy algorithm is *player symmetric* if its ranking function r does not depend on its first parameter, and we say that it is *object symmetric* if its ranking function r does not distinguish between sets of the same cardinality in its second parameter.

Lemma 5.6.1 *If \mathcal{A} is a player-symmetric greedy algorithm and a $c(n)$ -approximation whenever all declarations are single-minded, then for any declaration profile \mathbf{d} and allocation profile $\mathbf{y} = y_1, \dots, y_n$, $\sum_{i \in [n]} d_i(\mathcal{A}_i(\mathbf{d})) \geq \frac{1}{c(2n)} \sum_{i \in [n]} \theta_i(y_i, \mathbf{d}_{-i})$*

Proof: We define \mathbf{d}' as in Lemma 5.2.3. We then define \mathbf{d}'' by adding n additional bidders, $1', \dots, n'$, where $d_{i'}''$ is the single-minded declaration for set y_i at value $\theta_i(y_i, \mathbf{d}_{-i}) - \epsilon$. Player symmetry implies that $\mathcal{A}(\mathbf{d}') = \mathcal{A}(\mathbf{d}'')$ (meaning that each additional player is allocated \emptyset). Since we have $2n$ players, we conclude $SW(\mathcal{A}(\mathbf{d}'), \mathbf{d}^*) \geq \frac{1}{c} SW(\mathbf{y}, \mathbf{d}^*)$, yielding the desired result. \square

Applying Lemma 5.6.1 in place of Lemma 5.2.3, we can improve the statement of Theorems 5.4.2 so that the resulting prices of anarchy are improved from $c + 1$ to c , whenever algorithm \mathcal{A} is a c -approximation, but a $(c - 1)$ -approximation when agents are single-minded, and c is independent of n . This is the case, for example, in the standard greedy algorithm applied to cardinality-restricted combinatorial auctions.

We next show that if \mathcal{A} is both player-symmetric and object-symmetric, then we can again improve our bounds without additional assumptions on the performance when agents are single-minded.

Lemma 5.6.2 *If \mathcal{A} is player-symmetric, object-symmetric, and a $c(n, m)$ -approximation, then for any declaration profile \mathbf{d} and allocation profile $\mathbf{y} = y_1, \dots, y_n$,*

$$\sum_{i \in [n]} d_i(\mathcal{A}_i(\mathbf{d})) \geq \frac{1}{c(2n, 2m)} \sum_{i \in [n]} (\theta_i(y_i, \mathbf{d}_{-i}) + d_i(\mathcal{A}_i(\mathbf{d}))).$$

Proof: Consider an auction with an additional copy of each player and each object; write i' for the copy of agent i , and M' for the additional objects. The feasibility constraints for the new objects and agents are identical to those for the original objects and agents. Then \mathcal{A} is a $c(2n, 2m)$ approximation algorithm for this new problem instance.

Choose any $\epsilon > 0$. We define \mathbf{d}' as in Lemma 5.2.3. We then define \mathbf{d}'' by setting $d_{i'}'' = d_{i'}'$ and $d_{i''}''$ to be the single-minded declaration for set y_i at value $\theta_i(y_i, \mathbf{d}_{-i}) - \epsilon$. Finally, define \mathbf{d}''' by additionally adding a bid for the second copy of set $\mathcal{A}_i(\mathbf{d})$ by agent i for value $d_i(\mathcal{A}_i(\mathbf{d})) - \epsilon$. We then have $\mathcal{A}(\mathbf{d}''') = \mathcal{A}(\mathbf{d})$, but an alternative allocation gives y_i to each player i' , and the second copy of $\mathcal{A}_i(\mathbf{d})$ to agent i . The result then follows since \mathcal{A} is a $c(2n, 2m)$ approximation. \square

Applying Lemma 5.6.2 in place of Lemma 5.2.3, we can improve the statement of Theorems 5.4.2 so that the resulting price of anarchy is improved from $c+1$ to c whenever the algorithm is invariant with respect to the labels of the objects and bidders, and c does not depend on n or m .

We now give an example to show that these improved bounds are tight, even for pure Nash equilibria, for both $\mathcal{M}_1(\mathcal{A})$ and $\mathcal{M}_{crit}(\mathcal{A})$. That is, a pure Nash equilibrium can have approximation ratio as high as c for a c -approximate algorithm, where c is a constant, even if the algorithm is $(c-1)$ -approximate when we assume that all bidders are single-minded.

Example 5.6.3 *Consider a combinatorial auction with the additional requirement that each bidder can be given at most 2 objects. The standard greedy algorithm that allocates in order of value is a 3 approximation. This algorithm and problem are player and object*

symmetric, and furthermore this algorithm is a 2 approximation when agents are single-minded.

Consider the following valuation profile. There are 3 bidders and 3 objects, say $\{a, b, c\}$. Choose arbitrarily small $\epsilon > 0$; the valuations of the players are as in the following table.

<i>player</i>	<i>set</i>	<i>value</i>
1	$\{a, b\}$	$1 + 3\epsilon$
1	$\{c\}$	1
2	$\{a\}$	1
2	$\{b, c\}$	$1 + \epsilon$
3	$\{b\}$	1

The optimal solution gives each player their desired singleton at a value of 1, for a total welfare of 3. However, one pure nash equilibrium has each player bid truthfully, except having player 1 reduce his declared value for $\{a, b\}$ to the smallest value at which he will win it. This gives a total welfare of $1 + 3\epsilon$. So, for both $\mathcal{M}_1(\mathcal{A})$ and $\mathcal{M}_{crit}(\mathcal{A})$, the price of anarchy is at least 3 in both pure and mixed strategies.

5.7 Applications

We now describe some applications of our results to particular combinatorial allocation problems, resulting in mechanisms whose prices of anarchy improve on the approximation ratios of the best known incentive compatible algorithms.

Combinatorial Auctions The general combinatorial auction problem is defined by the feasibility constraint that no two allocations can intersect. Lehmann et al [64] show that the (non-adaptive) greedy allocation rule with $r(i, S, v) = \frac{v}{\sqrt{|S|}}$ achieves a $\sqrt{2m}$ approximation ratio for CAs. By Theorem 5.3.4, the deterministic first-price mechanism

for this algorithm has a $(\sqrt{2m} + O(m/e^{\sqrt{m}}))$ Bayesian price of anarchy. When agent types are correlated, Theorem 5.3.9 demonstrates that this mechanism has Bayesian price of anarchy at most $4\sqrt{2m}$. If we further assume that agents do not overbid, then the Bayesian price of anarchy for the deterministic critical-price mechanism is $(\sqrt{2m} + 1)$.

An alternative allocation rule, which can be implemented with a polynomial number of demand queries, was proposed by Mu'alem and Nisan [71]. This allocation rule combines a standard greedy algorithm with an allocation of all objects to a single bidder. By Theorem 5.5.2, this algorithm can also be implemented as a mechanism with $O(\sqrt{m})$ Bayesian price of anarchy.

Cardinality-restricted Combinatorial Auctions In the special case that players' desires are restricted to sets of size at most k , the non-adaptive greedy algorithm with $r(i, S, v) = v$ is k -approximate assuming single-minded agents. This translates to a $(k+1)$ approximate algorithm for general agents, which can be implemented as a mechanism with a $(k+1)$ price of anarchy assuming that agents do not overbid (by Theorem 5.4.2 with Lemma 5.6.1), or as a mechanism with a $k + O(k^2/e^k)$ Bayesian price of anarchy with no such assumption (by Theorem 5.3.4).

Multiple-Demand Unsplittable Flow Problem In the unsplittable flow problem (UFP), we are given an undirected graph with edge capacities. The objects are the edges, and each valuation function is such that agent i has some value $v(s, t)$ for being given a path from s to t . Each agent additionally specifies a fractional demand $d_i \in [0, 1]$ corresponding to a desired amount of flow to send along the given path. An allocation is feasible if the total allocated flow along each edge is no more than its capacity. Let B be the minimum edge capacity. A primal-dual algorithm, which is an adaptive greedy allocation rule, obtains an $O(m^{1/(B-1)})$ approximation for any $B > 1$ [18]. Theorem 5.3.4 implies that the first-price mechanism for this algorithm yields Bayesian price of anarchy of $O(m^{1/(B-1)})$.

Convex Bundle Auctions In a convex bundle auction, M is the plane \mathbb{R}^2 , and allocations must be non-intersecting compact convex sets. We suppose that agents declare valuation functions by making bids for such sets. Given such a collection of bids, the aspect-ratio, R , is defined to be the maximum diameter of a set divided by the minimum width of a set. A non-adaptive greedy allocation rule using a geometrically-motivated priority function yields an $O(R^{4/3})$ approximation [7]. Alternative greedy algorithms yield better approximation ratios for special cases, such as rectangles.

By Theorem 5.3.4, the deterministic first-price mechanism for this algorithm has a $O(R^{4/3})$ Bayesian price of anarchy. Since the allocation rule is non-adaptive, this bound holds when agent types are arbitrarily correlated.

Max-profit Unit Job Scheduling In this problem, each bidder has a job of unit time to schedule on one of multiple machines. A bidder has various windows of time of the form (release time, deadline, machine) in which his job could be scheduled, with a potentially different profit resulting from each window. The profits and windows are private information to each bidder. The goal of the mechanism is to schedule the jobs to maximize the total profit. The greedy algorithm that orders bids by value obtains a 3-approximation, and is symmetric with respect to agents and objects. Therefore, by Lemma 5.6.2, this greedy algorithm can be implemented as a mechanism that attains a price of anarchy of 3 assuming that bidders do not overbid. If we assume that agents will follow the weakly dominant set of strategies in which they do not overbid, $\mathcal{M}_{crit}(\mathcal{A})$ will always have a pure equilibrium and will have an price of anarchy of 3 in pure strategies.

Unlike the previous examples, an exact algorithm is known for the case of single-minded bidders [9], which uses dynamic programming and runs in time $O(n^7)$. Since this algorithm solves the problem optimally, it is incentive compatible. In this case, the greedy mechanism with price of anarchy 3 is appealing primarily due to its linear runtime and simple allocation rule.

Chapter 6

Repeated Combinatorial Auctions

In the previous chapter we analyzed the social welfare generated by a class of greedy combinatorial auction mechanisms when agents apply strategies at equilibrium. These results implicitly assume that agents are able to find strategies at equilibrium. However, as has been noted elsewhere [14, 41], there are a number of reasons to believe that an equilibrium assumption is often unrealistic for algorithmic mechanism design problems. First, equilibria are computationally difficult to find in general, and may not exist without the presence of agents who randomize over strategies for no reason other than to preserve the stability of the system. Even when equilibria exist and can be found, it is not clear that agents will always converge to an equilibrium under repeated play, or agree on which equilibrium to follow in a single-shot game.

In this chapter, we will focus our attention on alternative models of agent behaviour that weaken the equilibrium assumption, and have gained recent interest in the algorithmic game theory literature. These solution concepts explicitly model a *repeated-game* variant of combinatorial allocation problems, in which an auction problem is resolved multiple times with the same objects and bidders. Perhaps the most well-studied modern examples of repeated auctions are auctions for advertising spaces or slots [38], but this model applies also to bandwidth auctions (such as the FCC spectrum auction), airline

landing rights auctions [25], etc. In these settings a mechanism for the (one-shot) auction problem corresponds to a repeated game to be played by the agents. Even in settings where the auction is not explicitly repeated, we can view the repeated-game setting as modelling the process by which agents may (or may not) converge to an equilibrium via iteratively updating their strategy choices.

One may view a repeated auction as an extensive-form game, in which a strategy describes how to behave in each round given the outcomes of the previous rounds. One might then be tempted to study the Nash equilibria of this repeated-auction game. In general, however, it is a folk theorem that almost any outcome¹ can be implemented as an equilibrium of a repeated game, and these equilibria can be unintuitive and complex. For example, one player may threaten that unless a certain outcome occurs on a given round, then he will play in such a way to lower the utilities of the other agents (possibly to his own detriment) in subsequent rounds. Such threat strategies can cause the emergence of equilibrium outcomes that are unimplementable in the single-shot auction. For this reason, we do not study fully rational equilibria of the repeated auction game. Instead, we assume a form of limited rationality that attempts to capture natural bidding behaviour. We study two such models: external regret minimization and asynchronous best-response.

In the first model, agents can play arbitrary sequences of strategies for the repeated auction, under the assumption that they obtain low regret relative to the best fixed strategy in hindsight. More precisely, for each bidder, the difference between the average utility obtained by the bidder and the average utility that would have been obtained by the best single declaration in hindsight must tend to 0 as the number of auction rounds increases. These regret-minimizing bidders can be seen as agents that learn how to bid intelligently (relative to any fixed strategy benchmark) from the bidding history of past auction iterations. Note that we require no assumptions about the synchrony or

¹Specifically, one can implement any convex combination of outcomes in which each agent's expected utility is no less than the utility he could guarantee for himself regardless of how the other agents behave.

asynchrony of updates; arbitrary sets of agents can update their strategies concurrently. The regret-minimization assumption is motivated by the existence of simple, efficient algorithms that minimize external regret for optimization problems such as repeated auctions [57, 56]. Under this model, our goal is to design an auction mechanism that achieves an approximation to the optimal social welfare *on average* over sufficiently many rounds of the repeated auction. This is precisely the problem of designing a mechanism with bounded *price of total anarchy*, as introduced by Blum et al [14].

In the second model, we assume that agents choose strategies that are myopic best-responses to the current strategies of the other agents. Such bidding behaviour is best motivated in settings where agents update their declarations asynchronously (i.e. not concurrently). We model this behaviour as follows: on each auction round, an agent is chosen uniformly at random, and that agent will change his strategy to the current myopic best-response (or possibly maintain his current strategy if it constitutes a best-response). As in the regret-minimization model, our goal is to design auction mechanisms that achieve approximations to the best possible social welfare *on average* over sufficiently many auction rounds, with high probability over the random choices of bidders to update. This is closely related to the concept of the *price of (myopic) sinking*, as introduced by Goemans et al [41]. One method for bounding the price of sinking is to prove that there exists an equilibrium state that is reachable from any declaration profile by some polynomial-length sequence of best-response steps. This would imply that an equilibrium state would be reached with high probability after exponentially many steps. We do not take this approach, but rather prove the stronger result that the average social welfare obtained after a *polynomial* number of steps will approximate the optimal welfare with high probability.

Our high-level goal is to decouple computational issues from incentives issues. A full (and admittedly ambitious) solution in our domain would be a black-box conversion of a

given approximation algorithm into a mechanism that implements² the same approximation ratio, on average over sufficiently many auction rounds, given our model of bidder behaviour. The primary question raised in this chapter, which we partially address, is to what extent such implementations are possible.

As in Chapter 5, we design mechanisms that are based on a class of greedy approximation algorithms. Our goal is to convert a given approximation algorithm into a mechanism without loss in performance guarantees when agents behave in accordance to our model of rationality. What we find, loosely speaking, is that greedy approximation algorithms for combinatorial auction problems can be easily converted into mechanisms for regret-minimizing or myopic bidders.

Results Our first result is that any monotone greedy c -approximate algorithm can be implemented as a mechanism with price of total anarchy at most $c + 1$. Our mechanism is a black-box reduction from an algorithm for a one-shot auction iteration, and the same mechanism is applied each auction round. The form of our mechanism is very simple: on each round, it applies a simple modification to the bidders' declarations, then runs the approximation algorithm on the modified declarations and charges critical prices. This mechanism is quite similar to the mechanism $\mathcal{M}_{crit}(\mathcal{A})$ introduced in Chapter 5; the purpose of our modification is to reduce the complexity of the regret minimization problem for the agents. As in Chapter 5, our results make use of the property of strong loser-independence, which is satisfied by greedy algorithms.

Our implementation does not depend on the specific algorithms used by the agents to minimize their regret; only that their regret vanishes as the number of rounds increases. The rate of convergence to our approximation bound will depend on the rate at which the agents' regret vanishes.

We also show that our mechanism is robust against various perturbations of our equi-

²Throughout the paper we use the term “implement” in the economic sense of constructing a mechanism that obtains the desired properties when used by rational agents.

librium concept. First, we demonstrate that our mechanism is resilient to the presence of irrational agents, in the following sense. If each agent either applies regret-minimizing strategies or makes arbitrary declarations (but never declares more than his true value for a set), then the mechanism attains a $c + 1$ approximation to the optimal welfare *obtainable by the regret-minimizing bidders*. The no-overbidding assumption is necessary (as otherwise a irrational agent could bid arbitrarily high and prevent any welfare from being obtained) and motivated by viewing irrational players as not understanding how to participate intelligently in the auction and thus likely to bid conservatively.

We also show that our performance bound degrades gracefully if agents can only approximately minimize their regret. For $\gamma \geq 1$, if each agent is assumed to obtain a γ -approximation to the average utility of the single best strategy in hindsight, then our mechanism will obtain a $c + \gamma$ approximation to the optimal welfare (in addition to the additive error due to regret that vanishes as the number of rounds grows).

We then consider the best-response model, in which we focus specifically on the general combinatorial auction problem. We present a mechanism that implements an $O(s)$ approximation for cardinality-restricted combinatorial auctions where set allocations have size at most s , then extend this to a mechanism that implements an $O(\sqrt{m})$ approximation for general combinatorial auctions. We attain these approximation ratios with high probability after polynomially many auction rounds (in fact, only a slightly superlinear number of rounds).

We conjecture that the black-box reduction used to obtain a $c + 1$ approximation in the regret-minimization setting also implements an $O(c)$ approximation, on average over sufficiently many rounds, in the model of best-response bidders. We leave this conjecture as an open problem.

Our results require a mild game-theoretic assumption, which is that bidders will not apply strategies that are (strictly) dominated by easily-found alternatives. This is precisely the assumption of algorithmically undominated strategies, as introduced by

Babaioff et al [8]³. Additionally, the mechanism for best-response bidders applies a technique known in implementation theory as *virtual implementation*, where an alternative social choice rule is applied with vanishingly small probability [55]. We view this not as an introduction of randomness into the algorithm being implemented, but rather as the introduction of a trembling-hand consideration into the solution concept that encourages reasonable behaviour when best-response agents must distinguish between otherwise equally beneficial strategies.

6.1 Preliminaries

6.1.1 Repeated Auctions

We consider an instance of a combinatorial allocation problem that proceeds in rounds. The problem will be resolved by a direct revelation mechanism \mathcal{M} , say with allocation algorithm \mathcal{A} , which independently executes on each round of the auction. As usual, we will tend to write \mathbf{x} for the allocation rule associated with algorithm \mathcal{A} . We will make use of the definitions of *greedy* algorithms and *strongly loser-independent* algorithms from Chapter 5.

We assume that neither the agent types nor the mechanism changes between rounds of the auction. When the agents have types \mathbf{v} and $D = (\mathbf{d}^1, \mathbf{d}^2, \dots, \mathbf{d}^T)$ is a sequence of declared valuation profiles, we will write $SW_{\mathcal{A}}(D) = \frac{1}{T} \sum_t SW(\mathbf{x}(\mathbf{d}^t), \mathbf{v})$ for the average welfare obtained over all declarations in D . We will sometimes replace subscript \mathcal{A} by \mathcal{M} , in which case the social welfare is for the allocation rule of \mathcal{M} .

Declaration sequence $D = (\mathbf{d}^0, \mathbf{d}^1, \dots, \mathbf{d}^T)$ *minimizes external regret* for agent i if, for any fixed declaration d_i , $\sum_t u_i(d_i^t, \mathbf{d}_{-i}^t) \geq \sum_t u_i(d_i, \mathbf{d}_{-i}^t) + o(T)$. That is, the utility

³As discussed in Section 6.2, this assumption is not without loss of generality, but is intuitively motivated by the observation that the adoption of a dominated strategy corresponds to unnaturally risky bidding behaviour with no benefit.

of agent i approaches the utility of the optimal fixed strategy in hindsight.

Declaration sequence $D = (\mathbf{d}^0, \mathbf{d}^1, \dots, \mathbf{d}^T)$ is an instance of *response dynamics* if, for all $1 \leq t \leq T$, profiles \mathbf{d}^{t-1} and \mathbf{d}^t differ on the declaration of at most one player. Response dynamics D is an instance of *best-response dynamics* if, whenever \mathbf{d}^{t-1} and \mathbf{d}^t differ on the declaration of agent i , d_i^t maximizes agent i 's utility given the declarations of the other bidders. That is, $d_i^t \in \operatorname{argmax}_d \{u_i(d, \mathbf{d}_{-i}^t)\}$.

6.1.2 Regret Minimization

We now describe the concept of external regret minimization in further detail. The external regret of a sequence of declarations is the difference between the average utility of an agent (i.e. value of goods received minus payment extracted) and the maximum average utility that could have been obtained by a single fixed declaration made each round. An algorithm for generating declarations is regret-minimizing if its regret vanishes as a function of the number of auction rounds.

Our bounds on mechanism performance depend upon the agents actually minimizing their external regret, but how reasonable is it to assume that agents will do so? We recall that there is a simple and efficient algorithm for minimizing regret due to Kalai and Vempala [57] for minimizing regret, known as “follow the perturbed leader,” which requires time polynomial in the number of actions that can be taken by the agent. They also provide an improved algorithm that minimizes regret for a linear optimization problem, which runs in time polynomial in the problem dimension. These algorithms require access to an exact best-response oracle, which chooses the strategy that maximizes utility given the expected outcomes of each strategy. Kakade et al [56] show how to use a γ -approximate best response oracle to achieve a γ -approximation to the best fixed declaration in hindsight.

The regret-minimization mechanism we construct in this paper will reduce the strategy space for a bidder to an optimization problem over the space of possible allocations

to that bidder. Using a follow-the-perturbed-leader approach, an agent can therefore minimize regret in time proportional to the number of desired sets in his valuation function. If we assume the number of desired sets for each agent is polynomially small (e.g. in the model where agents provide these bids explicitly to the auction algorithm), then we conclude that agents can minimize regret efficiently. Alternatively, one could provide access to a best-response (or approximate best-response) oracle for the approximation algorithms being implemented, where the oracle must compute a best response to a distribution over possible declarations of the other agents. The ability to compute best-response choices is also a necessary component for the setting of best-response bidders, where the best-response assumption certainly requires that such strategies can be found efficiently.

In general, however, the space of possible allocations is exponentially large, and has exponential dimension as a linear problem. Thus, any efficient best-response oracle (to a distribution over possible declarations of the other bidders) must depend on the model by which agents access their own preferences and the details of the problem setting and algorithm being implemented. We leave the construction of such oracles to the designers of mechanisms for specific problem domains.

6.2 A Mechanism for Regret-Minimizing Bidders

In this section we prove that if agents avoid algorithmically dominated strategies and minimize external regret, then a strongly loser-independent monotone algorithm \mathcal{A} can be converted into a mechanism with almost no loss to its average approximation ratio over many rounds. The mechanism, $\mathcal{M}_{crit}^*(\mathcal{A})$, is described in Figure 6.1. Mechanism $\mathcal{M}_{crit}^*(\mathcal{A})$ proceeds by first simplifying the declaration given by each agent, then passing the simplified declarations to algorithm \mathcal{A} . The resulting allocation is paired with a payment scheme that charges critical prices.

<p>Mechanism $\mathcal{M}_{crit}^*(\mathcal{A})$:</p> <hr/> <p>Input: Declaration profile $\mathbf{d} = d_1, \dots, d_n$.</p> <ol style="list-style-type: none"> 1. $d' \leftarrow \text{SIMPLIFY}(\mathbf{d})$. 2. Allocate $\mathcal{A}(\mathbf{d}')$, charge critical prices.
<p>Procedure SIMPLIFY:</p> <hr/> <p>Input: Declaration profile $\mathbf{d} = d_1, \dots, d_n$.</p> <ol style="list-style-type: none"> 1. For each $i \in [n]$: 2. Choose $S_i \in \text{argmax}_S \{d_i(S)\}$, breaking ties in favour of smaller sets. 3. $d'_i \leftarrow (S_i, d_i(S_i))$. 4. Return (d'_1, \dots, d'_n).

Figure 6.1: A mechanism for regret-minimizing bidders, based on monotone algorithm \mathcal{A} .

The simplification process **SIMPLIFY** essentially converts any declaration into a single-minded declaration (and does not affect declarations that are already single-minded). We can therefore assume without loss of generality that agents always make single-minded declarations to this mechanism, as additional information is not used.⁴

Fix a particular combinatorial auction problem and type profile \mathbf{v} , and let \mathcal{A} be some monotone approximation algorithm. Since \mathbf{v} is fixed, we can think of a strategy for each agent i as a declaration $d_i \in V_i$. Let \mathbf{d} be a declaration profile; we suppose each d_i is a single-minded bid for set S_i (and, in general, we will write S_i for the set declared for in declaration d_i). We draw the following conclusion about the bidding choices of rational agents.

Lemma 6.2.1 *Let \mathcal{A} be a monotone approximation algorithm, and fix type profile \mathbf{v} . Then for each agent i , a single-minded declaration d_i for set S_i is an undominated strategy for mechanism $\mathcal{M}_{crit}^*(\mathcal{A})$ if and only if $d_i(S_i) = v_i(S_i)$.*

Proof: Fix some \mathbf{d}_{-i} and suppose d_i is a single-minded declaration for set S_i . On input (d_i, \mathbf{d}_{-i}) , mechanism $\mathcal{M}_{crit}^*(\mathcal{A})$ either allocates S_i or \emptyset to agent i . Thus agent i 's utility for declaring d_i , $u_i(d_i, \mathbf{d}_{-i})$, is $v_i(S_i) - \theta_i(S_i, \mathbf{d}_{-i})$ when $d_i(S_i) > \theta_i(S_i, \mathbf{d}_{-i})$, and 0 otherwise (where θ_i denotes critical prices with respect to $\mathcal{M}_{crit}^*(\mathcal{A})$). A declaration of $d_i(S_i) = v_i(S_i)$ therefore maximizes $u_i(d_i, \mathbf{d}_{-i})$ for all \mathbf{d}_{-i} .

Next suppose that $d_i(S_i) \neq v_i(S_i)$; we will show that d_i is not undominated. Let d_i' be the single-minded declaration for S_i at value $v_i(S_i)$. Suppose there is some \mathbf{d}_{-i} such that $\theta_i^{\mathcal{A}}(S_i, \mathbf{d}_{-i})$ lies strictly between $d_i(S_i)$ and $v_i(S_i)$. For simplicity we will assume such a \mathbf{d}_{-i} exists; handling the general case requires only a technical extension of notation⁵. Then if $d_i(S_i) < v_i(S_i)$, then $u_i(d_i', \mathbf{d}_{-i}) > 0 = u_i(d_i, \mathbf{d}_{-i})$. Otherwise, if $d_i(S_i) > v_i(S_i)$,

⁴We note, however, that this is not the same as assuming that agents are single-minded; our results hold for bidders with general private valuations.

⁵If $\theta_i^{\mathcal{A}}(S_i, \mathbf{d}_{-i})$ never lies between $d_i(S_i)$ and $v_i(S_i)$ for any \mathbf{d}_{-i} , then $\mathcal{M}_{\mathcal{A}}(d_i, \mathbf{d}_{-i}) = \mathcal{M}_{\mathcal{A}}(d_i', \mathbf{d}_{-i})$ for all \mathbf{d}_{-i} , so d_i and d_i' are equivalent strategies. We can therefore think of d_i as being “the same” as a single-minded declaration for S_i at value $v_i(S_i)$. We will ignore this technical issue for the remainder of the thesis, in the interest of clarity.

then $u_i(d_i', \mathbf{d}_{-i}) \geq 0 > u_i(d_i, \mathbf{d}_{-i})$. Thus, in either case, we have $u_i(d_i', \mathbf{d}_{-i}) > u_i(d_i, \mathbf{d}_{-i})$, and therefore declaration d_i' strictly dominates declaration d_i . \square

One implication of Lemma 6.2.1 is that the strategic choice of an agent participating in mechanism $\mathcal{M}_{\mathcal{A}}$ reduces to choosing a single set upon which to bid. On each round, we can think of agent i as choosing set S_i , which is the set he will attempt to win that round. Once S_i is chosen, an undominated declaration for agent i is determined: the single-minded declaration for S_i at value $v_i(S_i)$. Given that agent i chooses set S_i , his utility will be $v_i(S_i) - w_i$, where $w_i = \min\{v_i(S_i), \theta_i^{\mathcal{A}}(S_i, \mathbf{d}_{-i})\}$ is the price for set S_i , determined by the declarations of the other agents, capped at $v_i(S_i)$. Agents can then indeed apply the regret-minimization algorithm of Kalai and Vempala [57] to choose strategies that minimize external regret: they need simply maintain the expected payoff for the strategy of bidding upon each of their desired sets. The runtime of this algorithm is polynomial in the number of sets that agent i might bid upon; that is, if agent i has a type with k desired sets, then he can minimize his expected regret in time polynomial in k (i.e. the number of bids required to represent his valuation in the XOR bidding language).

We now proceed with bounding the social welfare obtained by $\mathcal{M}_{\mathcal{A}}$. Let \mathbf{y} be an optimal assignment for types \mathbf{v} . Suppose that $D = \mathbf{d}^1, \dots, \mathbf{d}^T$ is a sequence of declarations to our mechanism. The definition of regret minimization then immediately implies the following.

Lemma 6.2.2 *If agent i minimizes his external regret in bid sequence D , then*

$$\frac{1}{T} \sum_t (v_i(\mathcal{A}(\mathbf{d}^t)) + \theta_i^{\mathcal{A}}(y_i, \mathbf{d}_{-i}^t)) \geq v_i(y_i) - o(1).$$

Proof: Let d' be the single-minded declaration for set y_i at value $v_i(y_i)$. From the

definition of regret minimization,

$$\begin{aligned} \frac{1}{T} \sum_t u_i(d_i^t, \mathbf{d}_{-i}^t) &\geq \frac{1}{T} \sum_t u_i(d^t, \mathbf{d}_{-i}^t) - o(1) \\ &\geq \frac{1}{T} \sum_t (v_i(y_i) - \theta_i^{\mathcal{A}}(y_i, \mathbf{d}_{-i}^t)) - o(1) \\ &= v_i(y_i) - \frac{1}{T} \sum_t \theta_i^{\mathcal{A}}(y_i, \mathbf{d}_{-i}^t) - o(1). \end{aligned}$$

Since $u_i(d_i^t, \mathbf{d}_{-i}^t) \leq v_i(\mathcal{A}(\mathbf{d}^t))$ for all t , the result follows. \square

Assume now that algorithm \mathcal{A} is strongly loser-independent. We now recall from Lemma 5.2.3 in Chapter 5 that we can relate the value of the solution returned by an algorithm to the critical prices of the sets in an optimal solution. For completeness we restate this as Lemma 6.2.3.

Lemma 6.2.3 *If \mathcal{A} is a c -approximate strongly loser-independent algorithm, then for any type profile \mathbf{v} and optimal allocation profile \mathbf{y} , $\sum_{i \in [n]} v_i(\mathcal{A}_i(\mathbf{v})) \geq \frac{1}{c} \sum_{i \in [n]} \theta_i(y_i, \mathbf{v}_{-i})$.*

Furthermore, we recall from Lemma 5.2.2 that all greedy algorithms are strongly loser-independent. We are now ready to proceed with bounding the price of total anarchy for mechanism $\mathcal{M}_{crit}^*(\mathcal{A})$.

Theorem 6.2.4 *For any monotone greedy c -approximate algorithm \mathcal{A} , the mechanism $\mathcal{M}_{crit}^*(\mathcal{A})$ has price of total anarchy at most $c + 1$.*

Proof: Fix type profile \mathbf{v} , and let $D = \mathbf{d}^1, \dots, \mathbf{d}^T$ be a sequence of declarations in which all agents minimize external regret. Suppose that \mathbf{y} is an optimal allocation for \mathbf{v} . By Lemma 6.2.2, $\frac{1}{T} \sum_t (v_i(\mathcal{A}(\mathbf{d}^t)) + \theta_i(y_i, \mathbf{d}_{-i}^t)) \geq v_i(y_i) - o(1)$. Summing over all i , we have $\frac{1}{T} \sum_t \sum_i (v_i(\mathcal{A}(\mathbf{d}^t)) + \theta_i(y_i, \mathbf{d}_{-i}^t)) \geq SW_{OPT}(\mathbf{v}) - (n)(o(1))$. By Lemma 6.2.3, this implies $\frac{1}{T} \sum_t \sum_i (v_i(\mathcal{A}(\mathbf{d}^t)) + cd_i^t(\mathcal{A}(\mathbf{d}^t))) \geq SW_{OPT}(\mathbf{v}) - (n)(o(1))$. We know $d_i^t(\mathcal{A}(\mathbf{d}^t)) = v_i(\mathcal{A}(\mathbf{d}^t))$ for all i and t by Lemma 6.2.1, so we conclude $(c + 1)\frac{1}{T} \sum_t \sum_i v_i(\mathcal{A}(\mathbf{d}^t)) \geq$

$SW_{OPT}(\mathbf{v}) - (n)(o(1))$. Since the term hidden by the asymptotic notation vanishes with T and does not depend on n , we obtain the desired result. \square

The rate at which the welfare obtained by $\mathcal{M}_{crit}^*(\mathcal{A})$ converges to an average that is a $c + 1$ approximation to optimal depends on the rate of convergence of players' external regret to 0. The average welfare obtained after T rounds will have an additive loss of $(n)(r(T))$, where $r(T)$ is the average regret experienced by an agent after T rounds. Assuming that agents apply algorithms that minimize regret at a rate of $r(T) = o(1/\sqrt{T})$, which is attainable using the algorithm of Kalai and Vempala [57], the additive error term is at most a constant when T is at least quadratic in n .

We note that, since agents experience no regret at a pure Nash equilibrium, an immediate corollary to Theorem 6.2.4 is that mechanism $\mathcal{M}_{crit}^*(\mathcal{A})$ has price of anarchy at most $c + 1$. Indeed, Theorem 5.4.2 from the previous chapter can be extended to show that mechanism $\mathcal{M}_{crit}^*(\mathcal{A})$ has Bayesian price of anarchy at most $c + 1$. This result differs from Theorem 5.4.2 in that we do not require the additional assumption that agents do not overbid; only the weaker assumption that agents avoid dominated strategies.

6.2.1 Resilience to Irrational Strategies

Suppose that in addition to regret-minimizing agents, the auction participants include agents who do not follow rational strategies. The only restriction we impose on the behaviour of such agents is that they do not overbid on any set; that is, $d_i(S) \leq v_i(S)$ for any S . We can motivate this restriction either through our characterization of undominated strategies in Lemma 6.2.1, or by thinking of irrational players as not understanding how to participate rationally in the auction, and hence likely to be conservative in the way that they bid. Under this assumption, since Lemma 6.2.3 holds for any declaration profile, we easily obtain the following generalization of Theorem 6.2.4.

Proposition 6.2.5 *Let \mathcal{A} be a monotone greedy c -approximate algorithm. Suppose agents have types \mathbf{v} , D is a declaration sequence, and $N \subseteq [n]$ is a collection of agents that min-*

imize regret in D with respect to $\mathcal{M}_{crit}^*(\mathcal{A})$. Then if \mathbf{y} is an optimal allocation for the agents in N , and the remaining agents never bid more than their true values on any set in D , then $\frac{1}{T} \sum_t SW(\mathbf{x}(\mathbf{d}^t), \mathbf{v}) \geq \frac{1}{c+1} \sum_{i \in N} v_i(y_i) + |N|(o(1))$.

Proof: By Lemma 6.2.2 and summing over all $i \in N$,

$$\begin{aligned} & \sum_{i \in N} v_i(y_i) - |N|(o(1)) \\ & \leq \frac{1}{T} \sum_t \sum_{i \in N} (v_i(x_i(\mathbf{d}^t)) + \theta_i(y_i, \mathbf{d}_{-i}^t)) \\ & \leq \frac{1}{T} \sum_t \sum_{i \in [n]} (v_i(x_i(\mathbf{d}^t)) + \theta_i(y_i, \mathbf{d}_{-i}^t)). \end{aligned}$$

By Lemma 6.2.3, this implies $\frac{1}{T} \sum_t \sum_i (v_i(x_i(\mathbf{d}^t)) + cd_i^t(x_i(\mathbf{d}^t))) \geq \sum_{i \in N} v_i(y_i) - |N|(o(1))$.

The result then follows from the fact that $d_i^t(x_i(\mathbf{d}^t)) \leq v_i^t(x_i(\mathbf{d}^t))$ for all i . \square

6.2.2 Importance of Strong Loser-Independence

We note that the strong loser-independence property is necessary for Theorem 6.2.4, as the following example demonstrates.

Example 6.2.6 Consider an auction problem in which no agent can be allocated more than s objects, and moreover $M = A \cup B$ where $|A| = |B| = m/2$ and the mechanism must either allocate objects in A or objects in B , but not both. Consider the algorithm that takes the maximum over two solutions: a greedy assignment of subsets of A , and a greedy assignment of subsets of B . This algorithm is an $s + 1$ approximation.

Consider now an instance of the problem in which a single agent desires all of B with value 1, and each of $m/2$ agents desires a separate singleton in A with value $1 - \epsilon$. Suppose that the agent desiring B declares his valuation truthfully, but the other agents declare the zero valuation. On this input, the algorithm obtains social welfare 1, whereas $\frac{m}{2}(1 - \epsilon)$ is possible. However, this set of declarations forms a Nash equilibrium, and hence each agent has zero regret under this input profile. Thus, even if agents minimize

their regret, our mechanism may obtain a very poor approximation to the optimal social welfare over arbitrarily many auction rounds.

6.3 Best-Response Agents

In this section we consider the problem of designing mechanisms for agents that apply myopic best-response strategies asynchronously. Recall that in our model agents are chosen for update uniformly at random, one per round. In order to keep our exposition clear, we will make two additional assumptions about the nature of the best-response behaviour (which can be removed, as we discuss in Section 6.4). First, we will suppose that in the initial state every bidder makes the empty declaration \emptyset . Second, we suppose that if a bidder is chosen for update but cannot improve his utility, he will choose to maintain his previous strategy. These assumptions will simplify the process of characterizing best-response strategies of agents, and in particular the statement of Lemma 6.3.4 in the next section. It is possible to remove these assumptions, at the cost of a minor modification to the mechanisms we propose. We defer a more complete discussion to Section 6.4.

We begin our analysis of myopic bidders by considering mechanism $\mathcal{M}_{crit}^*(\mathcal{A})$ from Section 6.2, for any given monotone greedy algorithm \mathcal{A} . One might ask whether or not this mechanism converges to equilibrium under best-response dynamics. A simple example shows that this is not the case; mechanism $\mathcal{M}_{crit}^*(\mathcal{A})$ may not converge to a pure Nash equilibrium via best-response dynamics, even if pure equilibria exist.

Example 6.3.1 *Consider a combinatorial auction with 4 objects, say $\{a, b, c, d\}$, and 6 agents, under the feasibility constraint that each agent can receive at most 2 items. Let \mathcal{A} be the greedy allocation rule that allocates sets greedily by value. We consider an input instance given by the following set of true values (where the value for a set not listed is taken to be the maximum over its subsets).*

<i>player</i>	<i>set</i>	<i>value</i>
1	{ <i>a, b</i> }	4
1	{ <i>d</i> }	6
2	{ <i>a</i> }	2
2	{ <i>b, c</i> }	5
3	{ <i>c</i> }	4
4	{ <i>d</i> }	5

Suppose the auction is resolved by mechanism $\mathcal{M}_{crit}^*(\mathcal{A})$, and agents apply best-response dynamics. Agents 3 and 4 are single-minded and always maximize their utility by declaring truthfully. Agents 1 and 2 each have a strategic choice to make when bidding: which of their two desired sets should they bid upon? Note that once this decision is made, the way to bid is determined by Lemma 6.2.1 (i.e. bid truthfully for the desired set). It can be verified that from each of the resulting 4 possible declaration profiles, some player has incentive to change declaration. Thus best-response dynamics need not converge to an equilibrium.

The example above motivates a study of the *average* social welfare of $\mathcal{M}_{crit}^*(\mathcal{A})$, over many rounds of best-response dynamics. We conjecture that, on average, the best-response dynamics on mechanism $\mathcal{M}_{crit}^*(\mathcal{A})$ obtains an approximation to the optimal social welfare that is within a constant factor of the approximation ratio of the original algorithm \mathcal{A} .

Conjecture 6.3.2 *If \mathcal{A} is a monotone greedy c -approximate algorithm, then $\mathcal{M}_{crit}^*(\mathcal{A})$ has $O(c)$ price of (myopic) sinking.*

We leave the resolution of Conjecture 6.3.2 as an open problem. As partial progress, we construct alternative mechanisms that are more amenable to best-response analysis. These mechanisms are tailored specifically to the general combinatorial auction problem, and combinatorial auctions with cardinality-restricted sets.

6.3.1 The Approach

Our bound on the price of total anarchy of $\mathcal{M}_{crit}^*(\mathcal{A})$ in Theorem 6.2.4, as well as our price of anarchy bounds from the previous chapter, rely on a particular insight: if the social welfare of an auction outcome is low, then there must exist some agent i for whom the optimal assignment has a very low critical price. We used this to argue that an outcome with very low welfare cannot occur at equilibrium, since this agent i could improve his utility by pursuing his allocation in the optimal assignment.

The difficulty when extending this intuition to asynchronous best-response dynamics is that even if an agent can improve his utility by attempting to win some set for which the critical price is low, it may be that he has no chance to do so because he is not chosen to update his bid. Since each agent can expect to update his bid only once in every n rounds, our concern is that each agent spends most rounds wishing to make a bid that can greatly improve his utility, but cannot.

We address this difficulty in two steps. First, we modify our mechanism so that the social welfare is never much less than the sum of the bids of *all* players - even those that are not allocated their desired sets. We accomplish this by adding a restriction that a winning bid must be significantly larger than the sum of all bids it defeats. This implies that if agent i places a large bid on round t , then we can think of agent i as making a large contribution to the social welfare even if she does not win her bid. Second, we demonstrate that with high probability, in almost half of the rounds, either agent i places a large bid or else the critical price for his optimal allocation is high. Thus, even though agent i can modify his declaration only very infrequently, his (possibly indirect) contribution to the social welfare will still be large for approximately half of the rounds.

For the second step of the analysis, our primary tool will be the following probabilistic lemma, which pertains to any mechanism in a best-response setting. Suppose \mathcal{M} is a mechanism, and D is a sequence of best-response declarations for \mathcal{M} . For any \mathbf{d} , let $P_1(\mathbf{d}_{-i})$ be some property of \mathbf{d} that does not depend on d_i , and let $P_2(d_i)$ be some

property depending only on d_i .

Lemma 6.3.3 *Suppose that, for any \mathbf{d} such that d_i is a best-response to \mathbf{d}_{-i} , $\neg P_1(\mathbf{d}_{-i})$ implies $P_2(d_i)$. Then for all $\epsilon > 0$, if best-response dynamics is run for $T > \epsilon^{-1}n$ steps, there will be at least $(\frac{1}{2} - \epsilon)T$ steps t for which either $P_1(\mathbf{d}_{-i}^t)$ or $P_2(d_i^t)$ is true, with probability at least $1 - e^{-T\epsilon^2/32n}$.*

Proof: Our proof will make use of the *method of average bounded differences*. We will begin by giving a brief statement of this technique; see, for example, [35] for a more thorough treatment. Suppose that z_1, \dots, z_n are (not necessarily independent) random variables, and let f be real-valued function of z_1, \dots, z_n satisfying the property that, for each $i \in [n]$ and any two values a, a' that z_i can assume, there is a non-negative value c_i such that

$$|E[f|z_1, \dots, z_{i-1}; z_i = a] - E[f|z_1, \dots, z_{i-1}; z_i = a']| \leq c_i$$

where the expectations are with respect to the values of z_{i+1}, \dots, z_n . Then the method of average bounded differences states that $\Pr[f > \mathbf{E}[f] + \ell] \leq e^{-\ell^2/2c}$ for all $\ell > 0$, where $c = \sum_{i \in [n]} c_i^2$.

We now proceed with the proof of the lemma. We begin by defining a number of events, with respect to randomness in the instantiation of best-response dynamics. For the remainder of the proof we will not distinguish between events and indicator variables for those events; that is, an event Z will be associated with an indicator variable Z such that $Z = 1$ if and only if the event occurs.

For each $t \in [T]$, let B_i^t be the event that neither $P_1(\mathbf{d}_{-i}^t)$ nor $P_2(d_i^t)$ is true. Let A_i^t denote the event that $P_2(d_i^t)$ is true. Note that A_i^t and B_i^t are mutually exclusive. Consider the steps in which either A_i^t or B_i^t occurs: let $R = \{t: A_i^t \vee B_i^t\}$. For all $r \leq |R|$, let $t(r)$ denote the r th largest element of R . That is, $t(r)$ is the step at which either A_i^t or B_i^t is true for the r th time. Let C_r denote the event that $r \leq |R|$ and $B_i^{t(r)}$ is true. That is, C_r is the event that on the r th step in which either A_i^t or B_i^t occurs, it is B_i^t that

occurs.

We then note that $T - \sum_{r \leq T} C_r$ is precisely the number of steps for which either $P_1(\mathbf{d}_{-i}^t)$ or $P_2(d_i^t)$ is true. We therefore wish to show $Pr[\sum_{r \leq T} C_r > (\frac{1}{2} + \epsilon)T] < e^{-T\epsilon^2/32n}$, which implies our desired result.

Pick r and suppose that event C_r occurs. This implies that $B_i^{t(r)}$ occurs. With probability $\frac{1}{n}$ agent i is chosen for update on step $t(r) + 1$. Conditioning on that event, $A_i^{t(r)+1}$ must occur (by the assumption of the Lemma), and hence $t(r+1) = t(r) + 1$ and event C_{r+1} does not occur. We conclude $Pr[C_{r+1}|C_r] \leq (1 - 1/n)$.

Next suppose that event C_r does not occur; this implies that $A_i^{t(r)}$ occurs. With probability $(1 - \frac{1}{n})$ agent i is not chosen for update on step $t(r) + 1$. Conditioning on that event, $A_i^{t(r)+1}$ occurs (since P_2 depends only on the declaration of agent i , which does not change), and hence $t(r+1) = t(r) + 1$ and C_{r+1} does not occur. We conclude $Pr[C_{r+1}|\neg C_r] \leq 1/n$.

Let D_1, D_2, \dots, D_T be a random walk on $\{0, 1\}$ defined by $Pr[D_r|D_{r-1}] = (1 - 1/n)$, $Pr[D_r|\neg D_{r-1}] = 1/n$, and initial condition D_0 . Then our bounds above imply that $\sum_r C_r$ is stochastically dominated by $\sum_r D_r$, and hence $Pr[\sum_r C_r > (\frac{1}{2} + \epsilon)T] \leq Pr[\sum_r D_r > (\frac{1}{2} + \epsilon)T]$. It will therefore suffice to show that $Pr[\sum_{r \leq T} D_r > (\frac{1}{2} + \epsilon)T] < e^{-T\epsilon^2/32n}$.

The definition of D_r yields

$$\begin{aligned} Pr[D_r] &= \frac{1}{n}(1 - Pr[D_{r-1}]) + \left(1 - \frac{1}{n}\right) Pr[D_{r-1}] \\ &= \frac{1}{n} + \left(1 - \frac{2}{n}\right) Pr[D_r]. \end{aligned}$$

Solving this linear recurrence (with initial condition $D_0 \in \{0, 1\}$) yields

$$\begin{aligned} Pr[D_r] &= \frac{1}{2}(1 - (1 - 2/n)^r) + D_0(1 - 2/n)^r \\ &= \frac{1}{2} + (D_0 - \frac{1}{2})(1 - 2/n)^r. \end{aligned}$$

Linearity of expectation then implies

$$E\left[\sum_r D_r\right] = \frac{1}{2}T + \left(D_0 - \frac{1}{2}\right)\frac{n}{2}(1 - (1 - 2/n)^{T+1}).$$

From this we conclude that

$$E \left[\sum_r D_r \right] < \frac{1}{2}T + \frac{n}{4} \quad (6.1)$$

and moreover

$$\left| E \left[\sum_r D_r \middle| D_0 = 1 \right] - E \left[\sum_r D_r \middle| D_0 = 0 \right] \right| < \frac{n}{2}. \quad (6.2)$$

Let $k = T/n$ and define random variables F_1, \dots, F_k by $F_i = \sum_{r \in [in, (i+1)n-1]} D_r$. Note that $F_i \in [0, n]$ for all i . Note also that

$$\sum_r D_r = \sum_i F_i. \quad (6.3)$$

We would now like to apply the method of average bounded differences to random variables F_1, \dots, F_k and function $f = \sum_i F_i$. To do so, we must consider the expectation $\mathbf{E}[\sum_i F_i | F_1, \dots, F_{i-1}, F_i = \alpha]$ for $\alpha \in [0, n]$. But note that the influence of F_ℓ on the values of $F_{\ell+1}, \dots, F_k$ is captured entirely by the value of $D_{(\ell+1)n-1}$, and from (6.2) the influence of $D_{(\ell+1)n-1}$ on $\sum_{r=(\ell+1)n}^T D_r = \sum_{r=\ell+1}^k F_r$ is bounded by $\frac{n}{2}$. Since the value of F_ℓ also influences the sum $\sum_i F_i$ directly by at most n (due to its being included in the summation), we conclude that for all $\alpha, \alpha' \in [0, n]$,

$$\left| \mathbf{E} \left[\sum_j F_j \middle| F_1, \dots, F_{i-1}, F_i = \alpha \right] - \mathbf{E} \left[\sum_j F_j \middle| F_1, \dots, F_{i-1}, F_i = \alpha' \right] \right| \leq 3n/2.$$

Thus, by the method of average bounded differences (and recalling that $k = T/n$), we conclude that

$$Pr \left[\sum_j F_j > \mathbf{E} \left[\sum_j F_j \right] + (\epsilon/2)T \right] \leq e^{-(T\epsilon/2)^2/2(3n/2)^2k} < e^{-T\epsilon^2/18n}. \quad (6.4)$$

Our final step is to bound $\mathbf{E} \left[\sum_j F_j \right] + (\epsilon/2)T$ from the left-hand side of (6.4). Since $T > \epsilon^{-1}n$, we have from (6.3) and (6.1) that

$$\mathbf{E} \left[\sum_j F_j \right] + (\epsilon/2)T = \mathbf{E} \left[\sum_j D_j \right] + (\epsilon/2)T \leq \frac{1}{2}T + \frac{n}{4} + (\epsilon/2)T < \left(\frac{1}{2} + \epsilon \right) T.$$

Thus (6.4) implies

$$Pr \left[\sum_j F_j > \left(\frac{1}{2} + \epsilon \right) T \right] < e^{-T\epsilon^2/32n}$$

and the result follows. \square

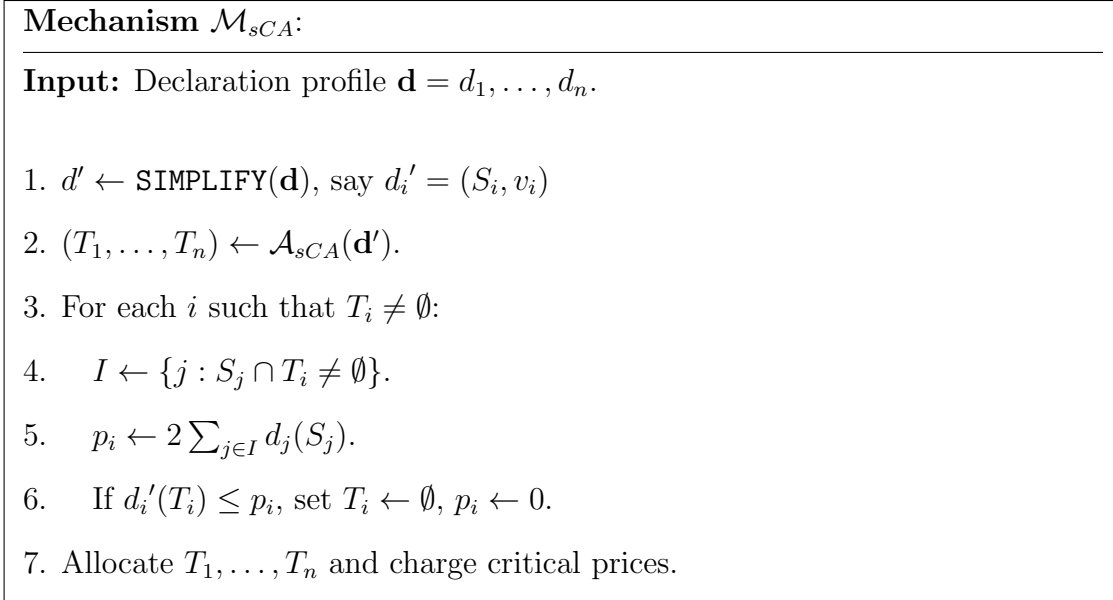


Figure 6.2: Mechanism \mathcal{M}_{sCA} , an implementation of greedy algorithm \mathcal{A}_{sCA} for the s -CA problem.

6.3.2 A Mechanism for s -CAs

Consider the s -CA problem, which is a combinatorial auction in which no agent can be allocated more than s objects. An algorithm that greedily assigns sets in descending order by value obtains an $(s+1)$ approximation.⁶ Call this algorithm \mathcal{A}_{sCA} . We will construct a mechanism \mathcal{M}_{sCA} based on \mathcal{A}_{sCA} ; it is described in Figure 6.2. This algorithm simplifies incoming bids (in the same way as $\mathcal{M}_{crit}^*(\mathcal{A})$) and runs algorithm \mathcal{A}_{sCA} to find a potential allocation. However, an additional condition for inclusion in the solution is imposed: the value declared for a set must be at least twice the sum of all bids for intersecting sets. Potential allocations that satisfy this condition are allocated, and the mechanism charges critical prices (that is, the smallest value at which an agent would be allocated their set by \mathcal{M}_{sCA} , which is not necessarily the same as the critical price for \mathcal{A}_{sCA}).

We note that since our mechanism implements a monotone algorithm and charges critical prices, Lemma 6.2.1 implies that undominated strategies for agent i involve choosing

⁶And an s approximation for single-minded declarations.

a set S_i and making a single-minded bid for S_i at value $v_i(S_i)$. We will therefore assume that agents bid in this way.

Suppose that \mathbf{d} is a declaration profile, where each d_i is single-minded for some set S_i . For any set T , define the set of bids intersecting T for i in \mathbf{d} , $I_i(\mathbf{d}, T)$, to be $\{j : j \neq i, S_j \cap T \neq \emptyset\}$. We also define $L_i(\mathbf{d}, T) = \{j : j \in I_i(\mathbf{d}, T), d_j(S_j) < v_i(T)\}$ to be the set of lower intersecting bids. We define the set of ancestors for agent i with respect to T and \mathbf{d} , $A_i(\mathbf{d}, T)$, recursively: it is the set of all lower intersecting bids, plus all ancestors of those lower intersecting bids. That is,

$$A_i(\mathbf{d}, T) = L_i(\mathbf{d}, T) \cup \bigcup_{j \in L_i(\mathbf{d}, T)} A_j(\mathbf{d}, S_j).$$

We say that \mathbf{d} is *separated for agent i* if $d_i(S_i) \geq 2 \sum_{j \in L_i(\mathbf{d}, S_i)} d_j(S_j)$ and \mathbf{d} is *separated* if it is separated for every bidder. Since an agent gains positive utility only if the declaration is separated for him, and since the initial state is the empty declaration profile (which is separated), we draw the following conclusion.

Lemma 6.3.4 *At each step of the best-response dynamics for mechanism \mathcal{M}_{sCA} , the declaration profile submitted by the agents will be separated.*

Proof: We will show that if declaration profile \mathbf{d} is separated, then it remains separated after a step of the best-response dynamics. The result then follows from our assumption that the initial state is the empty declaration (which is separated).

Suppose agent i is chosen to update his bid, say from d_i to d_i' . Let $\mathbf{d}' = (d_i', \mathbf{d}_{-i})$. If agent i cannot improve his utility then $\mathbf{d}' = \mathbf{d}$, so \mathbf{d}' is separated as required. Otherwise, he changes the set upon which he bids from, say, S_i to S_i' . Since $u_i(d_i', \mathbf{d}_{-i}) > 0$, it must be that $d_i(S_i) > 2 \sum_{j \in L_i(\mathbf{d}, S_i)} d_j(S_j)$. This implies that \mathbf{d}' is separated for agent i , and also implies that $d_i(S_i) > d_j(S_j)$ for all j such that $S_i \cap S_j \neq \emptyset$. Hence, for all $j \neq i$, we have $L_j(\mathbf{d}', S_j) \subseteq L_j(\mathbf{d}, S_j)$, and hence $\sum_{k \in L_j(\mathbf{d}', S_j)} d_k'(S_k) \leq \sum_{k \in L_j(\mathbf{d}, S_j)} d_k(S_k)$. Thus, since \mathbf{d} is separated for all $j \neq i$, \mathbf{d}' must be separated for each $j \neq i$ as well. \square

Motivated by Lemma 6.3.4, we will assume that declaration profiles are separated for

the remainder of this section. Under this assumption, the behaviour of mechanism \mathcal{M}_{sCA} simplifies in various fortuitous ways.

Proposition 6.3.5 *If \mathbf{d} is separated, then \mathcal{M}_{sCA} allocates S_i to agent i precisely when $d_i(S_i) > d_j(S_j)$ for all $j \in I_i(\mathbf{d}, S_i)$.*

Proof: Note first that $d_i(S_i) > d_j(S_j)$ for all $S_j \cap S_i \neq \emptyset$ if and only if $d_i(S_i) > \max_{j \in I_i(S_i, \mathbf{d})} d_j(S_j)$. If this is true, then S_i is allocated by $\mathcal{A}_{sCA}(\mathbf{d})$. Furthermore, we know $L_i(S_i, \mathbf{d}) = I_i(S_i, \mathbf{d})$, so (by separatedness) $d_i(S_i) > 2 \sum_{j \in I_i(S_i, \mathbf{d})} d_j(S_j)$ and hence S_i will be allocated by \mathcal{M}_{sCA} . On the other hand, if $d_i(S_i) \leq \max_{j \in I_i(S_i, \mathbf{d})} d_j(S_j)$, then certainly $d_i(S_i) \leq 2 \sum_{j \in I_i(S_i, \mathbf{d})} d_j(S_j)$ so S_i cannot be allocated by \mathcal{M}_{sCA} . \square

Proposition 6.3.6 *If \mathbf{d} is separated then $d_i(S_i) \geq \sum_{j \in A_i(\mathbf{d}, S_i)} d_j(S_j)$ for all i .*

Proof: By induction on the definition of $A_i(\mathbf{d}, T)$, we have that

$$\begin{aligned} d_i(S_i) &\geq 2 \sum_{j \in L_i(\mathbf{d}, S_i)} d_j(S_j) \\ &\geq \sum_{j \in L_i(\mathbf{d}, S_i)} d_j(S_j) + \sum_{j \in L_i(\mathbf{d}, S_i)} \sum_{k \in A_j(\mathbf{d}, S_j)} d_k(S_k) \\ &\geq \sum_{j \in A_i(\mathbf{d}, S_i)} d_j(S_j). \end{aligned}$$

\square

Proposition 6.3.7 *If \mathbf{d} is separated and mechanism \mathcal{M}_{sCA} allocates \emptyset to agent i , then there exists j such that $i \in A_j(\mathbf{d}, S_j)$ and \mathcal{M}_{sCA} allocates S_j to j .*

Proof: Suppose \mathcal{M}_{sCA} allocates \emptyset to agent i . Let $N = \{j : i \in A_j(\mathbf{d}, S_j)\}$, and choose $j \in \operatorname{argmax}_{j \in N} \{d_j(S_j)\}$. We claim that \mathcal{M}_{sCA} allocates S_j to j . If there exists k such that $d_k(S_k) > d_j(S_j)$ and $S_k \cap S_j \neq \emptyset$, then $j \in L_k(\mathbf{d}, S_k)$ and hence $k \in N$, contradicting the maximality of j . Also, since \mathbf{d} is separated, there cannot exist k such that $d_k(S_k) = d_j(S_j)$ and $S_k \cap S_j \neq \emptyset$. Thus $d_k(S_k) < d_j(S_j)$ for all k such that $S_k \cap S_j \neq \emptyset$, and hence \mathcal{M}_{sCA} allocates S_j to agent j by Proposition 6.3.5. \square

Lemma 6.3.8 *For all separated declarations \mathbf{d} , $SW_{\mathcal{M}_{sCA}}(\mathbf{d}) \geq \frac{1}{2} \sum_i d_i(S_i)$.*

Proof: By Lemma 6.3.6, if set S_i is allocated by \mathcal{M}_{sCA} then $d_i(S_i) \geq \sum_{j \in A_i(\mathbf{d}, S_i)} d_j(S_j)$. Let $N \subseteq [n]$ be the set of agents that receive non-empty sets in $\mathcal{M}_{sCA}(\mathbf{d})$. Lemma 6.3.7 implies that for all $j \notin N$ there is some $i \in N$ such that $j \in A_i(\mathbf{d}, S_j)$. Then

$$\begin{aligned} \sum_i d_i(S_i) &= \sum_{i \in N} d_i(S_i) + \sum_{i \notin N} d_i(S_i) \\ &\leq SW_{\mathcal{M}_{sCA}}(\mathbf{d}) + \sum_{i \in N} \sum_{j \in A_i(\mathbf{d}, S_i)} d_j(S_j) \\ &\leq SW_{\mathcal{M}_{sCA}}(\mathbf{d}) + \sum_{i \in N} d_i(S_i) \\ &= 2SW_{\mathcal{M}_{sCA}}(\mathbf{d}) \end{aligned}$$

as required. \square

We are now ready to invoke our high-level approach described above in order to bound the price of sinking for \mathcal{M}_{sCA} . Let \mathbf{y} be an optimal allocation with respect to the agents' true types \mathbf{v} . We will apply Lemma 6.3.3 with $P_1(\mathbf{d}_{-i})$ being the property that $\sum_{j \in I_i(\mathbf{d}, y_i)} d_j(S_j) < \frac{1}{4}v_i(y_i)$, and with $P_2(d_i)$ being the property that $d_i(S_i) \geq \frac{1}{2}v_i(y_i)$. We now show that P_1 and P_2 satisfy the properties required by Lemma 6.3.3.

Proposition 6.3.9 *If \mathbf{d} is separated and $\sum_{j \in I_i(\mathbf{d}, y_i)} d_j(S_j) < \frac{1}{4}v_i(y_i)$, then any utility-maximizing declaration for agent i , d_i' , will be a single-minded declaration for some S_i with $d_i'(S_i) \geq \frac{1}{2}v_i(y_i)$.*

Proof: Note that $\theta_i(y_i, \mathbf{d}_{-i}) = 2 \sum_{j \in I_i(\mathbf{d}, y_i)} d_j(S_j)$ in mechanism \mathcal{M}_{sCA} , so agent i would obtain utility at least $\frac{1}{2}v_i(y_i)$ by making a single-minded declaration for set y_i at value $v_i(y_i)$. Thus his utility-maximizing declaration must make at least this much utility, and therefore is a bid for some set S_i with $v_i(S_i) \geq \frac{1}{2}v_i(y_i)$. \square

For declaration profile \mathbf{d} , let G denote the set of agents i for which either $d_i(S_i) \geq \frac{1}{2}v_i(y_i)$ or $\sum_{j \in I_i(\mathbf{d}, y_i)} d_j(S_j) \geq \frac{1}{2}v_i(y_i)$. We think of this as the set of agents with ‘‘good’’

bids. We now bound the social welfare obtained by \mathcal{M}_{sCA} with respect to the optimal assignment to agents in G .

Lemma 6.3.10

$$SW_{\mathcal{M}_{sCA}}(\mathbf{d}) \geq \frac{1}{8(s+1)} \sum_{i \in G} v_i(y_i).$$

Proof: Let G_1 denote the set of agents for which $d_i(S_i) \geq \frac{1}{2}v_i(y_i)$, and let G_2 denote the set of agents for which $\sum_{j \in I_i(\mathbf{d}, y_i)} d_j(S_j) > \frac{1}{4}v_i(y_i)$, so that $G = G_1 \cup G_2$.

By Lemma 6.3.8, $SW_{\mathcal{M}_{sCA}}(\mathbf{d}) \geq \frac{1}{2} \sum_i d_i(S_i)$. We then have

$$SW_{\mathcal{M}_{sCA}}(\mathbf{d}) \geq \frac{1}{2} \sum_i d_i(S_i) \geq \frac{1}{2} \sum_{i \in G_1} d_i(S_i) \geq \frac{1}{4} \sum_{i \in G_1} v_i(y_i).$$

Also, $\sum_{i \in G_2} \sum_{j \in I_i(\mathbf{d}, y_i)} d_j(S_j) \geq \frac{1}{4} \sum_{i \in G_2} v_i(y_i)$. Since each S_j can intersect at most $|S_j| \leq s$ sets y_i , we know that each j appears in at most s different sets of the form $I_i(\mathbf{d}, y_i)$, and hence

$$\sum_{i \in G_2} \sum_{j \in I_i(\mathbf{d}, y_i)} d_j(S_j) \leq s \sum_j d_j(S_j).$$

We conclude that $s \sum_j d_j(S_j) \geq \frac{1}{4} \sum_{i \in G_2} v_i(y_i)$. Lemma 6.3.8 therefore implies that $(8s)SW_{\mathcal{M}_{sCA}}(\mathbf{d}) \geq \sum_{i \in G_2} v_i(y_i)$.

We conclude that $(8s+4)SW_{\mathcal{M}_{sCA}}(\mathbf{d}) \geq \sum_{i \in G_1} d_i(S_i) + \sum_{i \in G_2} d_i(S_i)$, which implies the desired result. \square

We are now ready to bound the average social welfare of our mechanism, over sufficiently many rounds, with respect to the approximation factor of algorithm \mathcal{A} .

Theorem 6.3.11 *Choose $\epsilon > 0$ and suppose $D = d^1, \dots, d^T$ is an instance of best-response dynamics with random player order, where agents play undominated strategies, and $T > \epsilon^{-1}n$. Then*

$$SW_{\mathcal{M}_{sCA}}(D) \geq \left(\frac{1-2\epsilon}{16(s+1)} \right) SW_{OPT}(\mathbf{v})$$

with probability at least $1 - ne^{-T\epsilon^2/32n}$.

Proof: Let G_t be the set of agents G from Lemma 6.3.10 on step t (i.e. with respect to declaration \mathbf{d}^t). Lemma 6.3.3 and Proposition 6.3.9 together imply that each agent i will be in G_t for at least $(\frac{1}{2} - \epsilon)T$ values of t , with probability at least $1 - e^{-T\epsilon^2/32n}$. The union bound then implies that this occurs for every agent with probability at least $1 - ne^{-T\epsilon^2/32n}$. Conditioning on the occurrence of this event, Lemma 6.3.10 implies

$$\begin{aligned} SW_{\mathcal{M}_{sCA}}(D) &= \frac{1}{T} \sum_t SW_{\mathcal{M}_{sCA}}(\mathbf{d}^t) \\ &\geq \frac{1}{8(s+1)T} \sum_t \sum_{i \in G_t} v_i(y_i) \\ &\geq \frac{1}{8(s+1)T} \sum_i T \left(\frac{1}{2} - \epsilon \right) v_i(y_i) \\ &\geq \left(\frac{1 - 2\epsilon}{16(s+1)} \right) SW_{OPT}(\mathbf{v}) \end{aligned}$$

which implies the required bound. \square

If we take ϵ to be a small constant and assume $T = \Omega(n^{1+\delta})$ for some $\delta > 0$, we conclude that $SW_{\mathcal{M}_{sCA}}(D) > \frac{1}{O(s)} SW_{opt}(\mathbf{v})$ with high probability. Thus \mathcal{M}_{sCA} implements an $O(s)$ approximation to the s -CA problem for best-response bidders over sufficiently many rounds. In other words, \mathcal{M}_{sCA} has $O(s)$ price of (myopic) sinking.

6.3.3 A Mechanism for General CAs

Consider the following algorithm for the general CA problem: try greedily assigning sets, of size at most \sqrt{m} , by value; return either the resulting solution or the allocation that gives all items to a single agent, whichever generates more welfare. This algorithm is known to be a $O(\sqrt{m})$ approximation for the general CA problem [71]. We will construct a mechanism \mathcal{M}_{CA} based on this algorithm; it is described in Figure 6.3. The mechanism \mathcal{M}_{CA} essentially implements two copies of \mathcal{M}_{sCA} (as described in the previous section): one for sets of size at most \sqrt{m} (which we will call $\mathcal{M}_{\sqrt{m}CA}$), and one for allocating all objects to a single bidder. It then returns whichever of the two solutions yields the largest social welfare. We add one additional modification: with vanishingly small probability

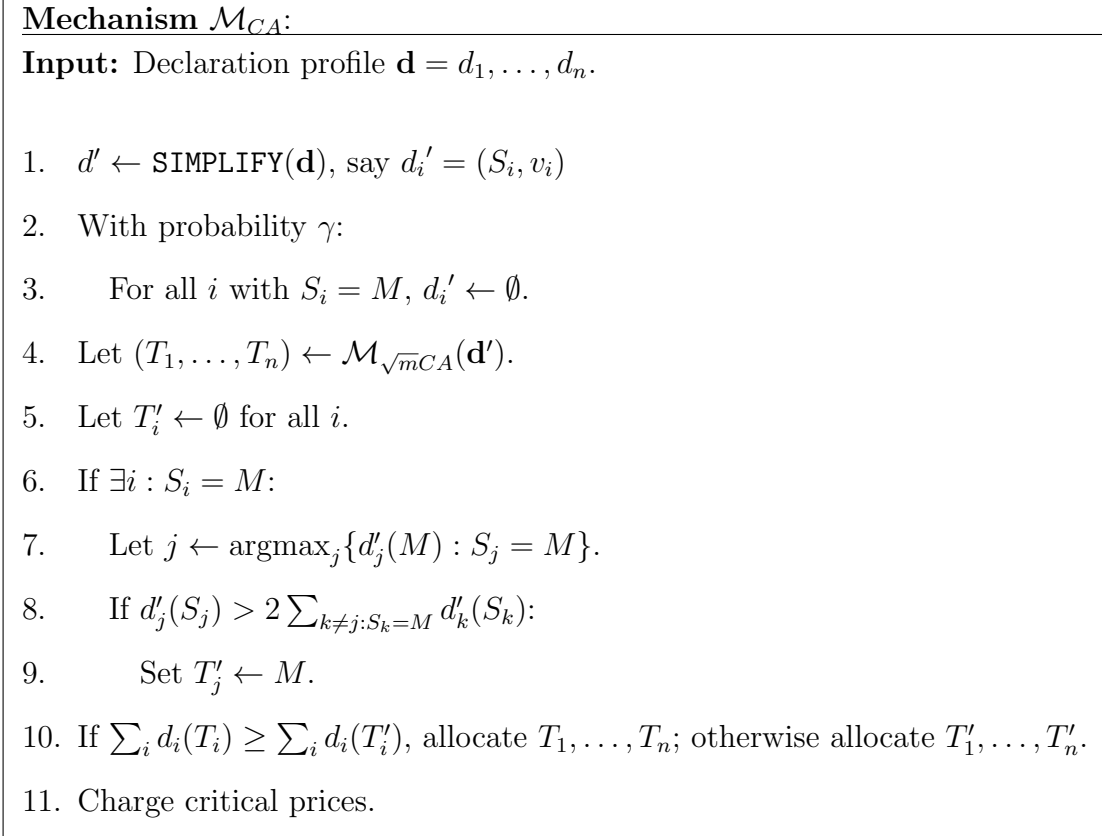


Figure 6.3: Mechanism \mathcal{M}_{CA} , a best-response implementation of a greedy algorithm for the CA problem. Parameter $\gamma > 0$ is an arbitrarily small positive constant. Note $\mathcal{M}_{\sqrt{m}CA}$ is \mathcal{M}_{sCA} from Figure 6.2 with $s = \sqrt{m}$.

γ , \mathcal{M}_{CA} ignores bids for M and behaves as $\mathcal{M}_{\sqrt{m}CA}$. The purpose of this modification is to encourage agents to bid on small sets, even when the presence of a high-valued bid for a large set would seem to indicate that bidding on small sets is fruitless.

The analysis of the average social welfare obtained by \mathcal{M}_{CA} closely follows the analysis for \mathcal{M}_{sCA} . Our high-level approach is to apply this analysis twice: once for allocations of sets of size at most \sqrt{m} , and once for allocations of all objects to a single bidder. The primary complicating factor is that the bidding choice of an agent may be influenced by the mechanism's choice of whether or not to allocate M to a single bidder; this can be handled by a careful analysis of utility-maximizing declarations. The final result is the following.

Theorem 6.3.12 *Choose $\epsilon > 0$ and suppose $D = d^1, \dots, d^T$ is an instance of best-response dynamics with random player order, where agents play undominated strategies, and $T > \epsilon^{-1}n$. Then*

$$SW_{\mathcal{M}_{CA}}(D) \geq \left(\frac{1 - 2\epsilon}{O(\sqrt{m})} \right) SW_{opt}(\mathbf{v})$$

with probability at least $1 - 2ne^{-T\epsilon^2/32n}$.

Note first that Lemma 6.2.1 applies to \mathcal{M}_{CA} for the same reason as \mathcal{M}_{sCA} , so we will assume that all declarations in an instance of best-response dynamics are single-minded declarations, in which each d_i is a bid for some set S_i at value $v_i(S_i)$. Also, we can assume that all single-minded declarations are for sets that are either M or of size at most \sqrt{m} , since these are the only sets allocated by \mathcal{M}_{CA} .

Recall the definitions of I_i , L_i , and A_i our analysis of \mathcal{M}_{sCA} . We will define these similarly, though we will separate bids for M from bids for sets of size at most \sqrt{m} . That is, if \mathbf{d} is a declaration where $|S_i| \leq \sqrt{m}$, then $I_i(\mathbf{d}, S_i) = \{j : S_j \cap S_i \neq \emptyset, |S_j| \leq \sqrt{m}\}$. If, on the other hand, $S_i = M$, then $I_i(\mathbf{d}, S_i) = \{j : S_j = M\}$ (which is the set of bids for M that intersect S_i). We extend the definitions for L_i and A_i in a similar way.

As in Section 6.3.2, we say that \mathbf{d} is *separated* for agent i if $d_i(S_i) \geq 2 \sum_{j \in L_i(\mathbf{d}, S_i)} d_j(S_j)$, and that \mathbf{d} is separated if it is separated for all agents. That is, \mathbf{d} is separated if the bids for sets of size at most \sqrt{m} are separated in the sense of \mathcal{M}_{sCA} , and the bids for M are separated in the sense of \mathcal{M}_{sCA} .

Lemma 6.3.4 applies to \mathcal{M}_{CA} for the same reasons as \mathcal{M}_{sCA} , so we will assume that all declarations in an instance of best-response dynamics are separated.

Fix some separated declaration profile \mathbf{d} . We now wish to characterize the utility-maximizing declarations of an agent i in order to obtain a variant of Proposition 6.3.9. The complicating factor is that we cannot consider bids for M and bids for small sets separately: bids on small sets can influence the critical price for a bid on M , and vice-versa, via the mechanism's choice of whether to allocate M to a single bidder or not on line 10. We begin with the simpler case of declarations for set M .

Lemma 6.3.13 *If $SW_{\mathcal{M}_{CA}}(\emptyset, \mathbf{d}_{-i}) < \frac{1}{8}v_i(M)$, then the utility-maximizing bid for agent i , d_i , sets $d_i(S_i) \geq \frac{1}{2}v_i(M)$ for some S_i .*

Proof: We claim that $\theta_i(M, \mathbf{d}_{-i}) \leq \frac{1}{2}v_i(M)$. To prove this we must show that a bid of $\frac{1}{2}v_i(M)$ for M would cause agent i to be allocated set M , which is true if and only if $\frac{1}{2}v_i(M) \geq 2 \sum_{j \in A_i(\mathbf{d}, M)} d_j(M)$ and $SW_{\mathcal{M}_{\sqrt{m}CA}}(\mathbf{d}) \leq \frac{1}{2}v_i(M)$.

For the first requirement, observe that since $SW_{\mathcal{M}_{CA}}(\emptyset, \mathbf{d}_{-i}) < \frac{1}{8}v_i(M)$ and \mathbf{d} is separated, it must be that in declaration profile \mathbf{d}_{-i} no agent (other than i) bid for M at a value greater than $\frac{1}{8}v_i(M)$. This implies, by separatedness, that the sum of all bids for M (by agents other than i) must be at most $\frac{1}{4}v_i(M)$, so $\frac{1}{2}v_i(M) \geq 2 \sum_{j \in A_i(\mathbf{d}, M)} d_j(M)$ as we required. For the second requirement, we note that the social welfare obtained by $\mathcal{M}_{\sqrt{m}CA}$ can be at most $SW_{\mathcal{M}_{CA}}(\emptyset, \mathbf{d}_{-i}) < \frac{1}{8}v_i(M)$. Thus $\theta_i(M, \mathbf{d}_{-i}) \leq \frac{1}{2}v_i(M)$ as claimed.

This implies that agent i can make a utility of $\frac{1}{2}v_i(M)$ by bidding on set M . Thus his utility-maximizing declaration must be a bid for a set S_i with $v_i(S_i) \geq \frac{1}{2}v_i(M)$. \square

We next consider allocations of small sets. Let \mathbf{y} be an optimal allocation of sets of size at most \sqrt{m} .

Lemma 6.3.14 *If $\sum_{j \in I_i(\mathbf{d}, y_i)} d_j(S_j) < \frac{1}{4}v_i(y_i)$, then the utility-maximizing bid for agent i , d_i , sets $d_i(S_i) \geq \frac{1}{2}v_i(y_i)$ for some S_i .*

Proof: For ease of notation we will write $SW_{\sqrt{m}}$ for $SW_{\mathcal{M}_{\sqrt{m}CA}}$ in this proof. We begin with a technical lemma that bounds the effect of a single agent's bid on the social welfare obtained by \mathcal{M}_{CA} .

Claim 6.3.15 *Suppose \mathbf{d} is separated and $\mathcal{M}_{\sqrt{m}CA}(\mathbf{d})$ allocates S_i to agent i . Then*

$$d_i(S_i) - \sum_{j \in L_i(\mathbf{d}, S_i)} d_j(S_j) \leq SW_{\sqrt{m}}(d_i, \mathbf{d}_{-i}) - SW_{\sqrt{m}}(\emptyset, \mathbf{d}_{-i}) \leq d_i(S_i).$$

Proof: Let N be the set of agents allocated a non-empty set by $\mathcal{M}_{\sqrt{m}CA}$ on input $(\emptyset, \mathbf{d}_{-i})$, and let N' be the set of agents allocated to by $\mathcal{M}_{\sqrt{m}CA}$ on input (d_i, \mathbf{d}_{-i}) . By Proposition 6.3.5, N consists of all $j \neq i$ for which $d_j(S_j) > d_k(S_k)$ for all $k \notin \{i, j\}$ such that $S_k \cap S_j \neq \emptyset$. Furthermore, N' consists of i plus all agents $j \in N$ such that $S_i \cap S_j \neq \emptyset$. That is, $N' = \{i\} \cup (N \setminus L_i(\mathbf{d}, S_i))$. From this we conclude that

$$SW_{\sqrt{m}}(\mathbf{d}) - SW_{\sqrt{m}}(\emptyset, \mathbf{d}_{-i}) = d_i(S_i) - \sum_{j \in N \cap L_i(\mathbf{d}, S_i)} d_j(S_j).$$

The claim then follows from noting that

$$d_i(S_i) \geq d_i(S_i) - \sum_{j \in N \cap L_i(\mathbf{d}, S_i)} d_j(S_j) \geq d_i(S_i) - \sum_{j \in L_i(\mathbf{d}, S_i)} d_j(S_j).$$

□

We now proceed with the proof of Lemma 6.3.14. Suppose for contradiction that the utility-maximizing declaration d_i is for a set S_i with $v_i(S_i) < v_i(y_i)/2$. Note, then, that $S_i \neq M$, since $v_i(M) \geq v_i(y_i)$. Let d_i' be the declaration for set y_i at value $v_i(y_i)$.

Let $x = \max_{j: S_j = M} d_j(M)$ be the maximum amount that any agent bids on M in \mathbf{d}_{-i} . Then \mathcal{M}_{CA} will allocate to agent i on declaration d_i only if $SW_{\sqrt{m}CA}(d_i, \mathbf{d}_{-i}) \geq x$. The utility obtained by agent i on declaration d_i is therefore at most $SW_{\sqrt{m}CA}(d_i, \mathbf{d}_{-i}) - x$. Similarly, the utility obtained by agent i on declaration d_i' is at most $SW_{\sqrt{m}CA}(d_i', \mathbf{d}_{-i}) - x$.

Now, from Claim 6.3.15, we know

$$\begin{aligned} SW_{\sqrt{m}CA}(d_i, \mathbf{d}_{-i}) &\leq SW_{\sqrt{m}CA}(\emptyset, \mathbf{d}_{-i}) + v_i(S_i) \\ &< SW_{\sqrt{m}CA}(\emptyset, \mathbf{d}_{-i}) + v_i(y_i)/2 \\ &\leq SW_{\sqrt{m}CA}(\emptyset, \mathbf{d}_{-i}) + v_i(y_i) - \sum_{j \in L_i(\mathbf{d}, y_i)} d_j(S_j) \\ &\leq SW_{\sqrt{m}CA}(d_i', \mathbf{d}_{-i}) \end{aligned} \tag{6.5}$$

where the first and last inequalities are due to Claim 6.3.15, and the third is from the assumptions of the lemma.

For the remainder of this proof we will write $\theta_i^{\mathcal{M}}(S, \mathbf{d}_{-i})$ for the critical price of set S in mechanism \mathcal{M} . We introduce this notation so that we can compare critical prices of mechanisms $\mathcal{M}_{\sqrt{m}CA}$ and \mathcal{M}_{CA} .

First, in the event that mechanism \mathcal{M}_{CA} chooses to ignore all bids for M on line 3 (which occurs with probability $\gamma > 0$), note that the mechanism will behave as $\mathcal{M}_{\sqrt{m}CA}$. Since the assumption of the lemma implies that $\theta_i^{\mathcal{M}_{\sqrt{m}CA}}(y_i, \mathbf{d}_{-i}) < v_i(y_i)/2$, agent i will receive utility greater than $v_i(y_i)/2$ by declaring d_i' subject to this event, which dominates the utility obtained by declaring d_i . Thus, to show that the expected utility of declaring d_i' is greater than that of d_i , it will be sufficient to argue that the utility of d_i' is at least that of d_i in the event that \mathcal{M}_{CA} does not execute line 3.

We now wish to analyze $\theta_i^{\mathcal{M}_{CA}}(y_i, \mathbf{d}_{-i})$, which could be larger than $\theta_i^{\mathcal{M}_{\sqrt{m}CA}}(y_i, \mathbf{d}_{-i})$ due to the possibility of allocating M to a single agent. We consider cases based on the value of $\theta_i^{\mathcal{M}_{CA}}(y_i, \mathbf{d}_{-i})$.

Case 1: $\theta_i^{\mathcal{M}_{CA}}(y_i, \mathbf{d}_{-i}) \leq v_i(y_i)/2$. In this case agent i obtains utility at least $v_i(y_i)/2$ by bidding upon set y_i , which is greater than the utility obtained by d_i (which can be at most $v_i(S_i) < v_i(y_i)/2$), a contradiction.

Case 2: $v_i(y_i)/2 < \theta_i^{\mathcal{M}_{CA}}(y_i, \mathbf{d}_{-i}) < v_i(y_i)$. In this case we have that $\theta_i^{\mathcal{M}_{CA}}(y_i, \mathbf{d}_{-i}) > \theta_i^{\mathcal{M}_{\sqrt{m}CA}}(y_i, \mathbf{d}_{-i})$. This corresponds to the case in which $0 < SW_{\sqrt{m}CA}(d_i', \mathbf{d}_{-i}) - x < v_i(y_i)/2$, recalling that $x = \max_{j: S_j=M} d_j(M)$. The utility generated by declaration d_i' is therefore precisely $SW_{\sqrt{m}CA}(d_i', \mathbf{d}_{-i}) - x$. But recall that the utility generated by declaration d_i is at most $SW_{\sqrt{m}CA}(d_i, \mathbf{d}_{-i}) - x$, which by (6.5) is less than $SW_{\sqrt{m}CA}(d_i', \mathbf{d}_{-i}) - x$, a contradiction.

Case 3: $\theta_i^{\mathcal{M}_{CA}}(y_i, \mathbf{d}_{-i}) > v_i(y_i)$. In this case, $SW_{\sqrt{m}CA}(d_i', \mathbf{d}_{-i}) < x$. By (6.5), we also have $SW_{\sqrt{m}CA}(d_i, \mathbf{d}_{-i}) < x$. Thus, for both d_i and d_i' , the generated utility will be 0. Thus $u_i(d_i', \mathbf{d}_{-i}) \geq u_i(d_i, \mathbf{d}_{-i})$, which contradicts the maximality of d_i (recalling that it is sufficient to show that the utility of d_i' is at least that of d_i , due to the possibility of the event that \mathcal{M}_{CA} executes line 3).

We have thus reached a contradiction in all cases, completing the proof. \square

We are now ready to complete the proof of Theorem 6.3.12, in a manner similar to Theorem 6.3.11.

Proof of Theorem 6.3.12 :

Since each \mathbf{d}^t is separated, it is clear that $SW_{\mathcal{M}_{CA}}(\mathbf{d}^t) \geq d_i^t(S_i)$ for all t . Applying Lemma 6.3.3 to Lemma 6.3.13 therefore implies that at for least $(\frac{1}{2} - \epsilon)T$ steps of D , $SW_{\mathcal{M}_{CA}}(\mathbf{d}^t) \geq \frac{1}{8}v_i(M)$. We conclude that, with probability at least $1 - ne^{-T\epsilon^2/32n}$, $SW_{\mathcal{M}_{sCA}}(D) \geq (\frac{1}{2} - \epsilon)\frac{1}{8} \max_i v_i(M)$. Let \mathbf{y} be the optimal allocation in which each non-empty set has size at least \sqrt{m} . Then $SW(\mathbf{y}, \mathbf{v}) \leq \sqrt{m} \max_i v_i(M)$, since there can be at most \sqrt{m} non-empty sets in \mathbf{y} . We therefore conclude that

$$SW_{\mathcal{M}_{sCA}}(D) \geq \frac{1 - 2\epsilon}{16\sqrt{m}} SW(\mathbf{y}, \mathbf{v}).$$

Now let \mathbf{z} be the optimal allocation of sets of size at most \sqrt{m} . For each t let G_t be the set of agents for which either $d_i^t(S_i^t) \geq v_i(z_i)/2$ or $\sum_{j \in I_i(\mathbf{d}^t, S_i^t)} d_j(S_j^t) \geq \frac{1}{4}v_i(z_i)$. Applying Lemma 6.3.3 to Lemma 6.3.13, we see that for each agent i appears in G_t for at least $(\frac{1}{2} - \epsilon)T$ time steps, with probability at least $1 - ne^{-T\epsilon^2/32n}$. The same argument as in the proof of Lemma 6.3.10 and Theorem 6.3.11 then demonstrates that, with probability at least $1 - ne^{-T\epsilon^2/32n}$,

$$SW_{\mathcal{M}_{sCA}}(D) \geq \frac{1 - 2\epsilon}{16\sqrt{m}} SW(\mathbf{z}, \mathbf{v}).$$

Taking the union bound over the events described above, and noting that $SW_{OPT}(\mathbf{v}) \leq SW(\mathbf{z}, \mathbf{v}) + SW(\mathbf{y}, \mathbf{v})$, we conclude that, with probability at least $1 - 2ne^{-T\epsilon^2/32n}$,

$$SW_{\mathcal{M}_{sCA}}(D) \geq \frac{1 - 2\epsilon}{32\sqrt{m}} SW_{OPT}(\mathbf{v})$$

as required. \square

We conclude that mechanism \mathcal{M}_{CA} implements an $O(\sqrt{m})$ approximation to the combinatorial auction problem for best-response bidders, with high probability, whenever $T = \Omega(n^{1+\delta})$ for $\delta > 0$.

6.4 Removing Assumptions

Recall that in our model of best-response dynamics, we assumed that in the initial state every bidder makes the empty declaration \emptyset , and that if a bidder is chosen for update but cannot improve his utility, he will choose to maintain his previous strategy. We used these assumptions to argue that agents make only separated declarations when participating in mechanisms \mathcal{M}_{sCA} and \mathcal{M}_{CA} .

These assumptions can be removed as follows. We can modify mechanisms \mathcal{M}_{sCA} and \mathcal{M}_{CA} so that, with vanishingly small probability, an alternative allocation rule is used. This alternative rule chooses an agent at random, and assigns him all objects at no cost *as long as the input declaration is separated for that agent*. Thus, any separated declaration by agent i results in positive expected utility. Then, since any non-separated declaration by an agent results in a utility of 0 for that agent, it must be that the utility-maximizing declaration by any agent must be a separated declaration.

It follows that after each bidder is chosen at least once for update, and every step thereafter, the input declaration will be separated. Thus, with high probability, every declaration after $O(n \log n)$ steps will be separated (by the coupon-collector problem). Lemma 6.3.4 will therefore hold after $O(n \log n)$ steps of best-response dynamics, with high probability. The remainder of the analysis can then proceed without change.

Chapter 7

Truthful Priority Algorithms

Whether it is possible to design deterministic truthful mechanisms for combinatorial auction problems with performance approaching that of the best-known approximation algorithms is one of the primary open questions in algorithmic mechanism design. For example, without strategic considerations, one can obtain an $O(\min\{n, \sqrt{m}\})$ approximation to the general combinatorial auction problem with n bidders and m objects with a conceptually simple greedy algorithm [64], and this is the best possible under standard complexity assumptions [49, 87]. However, no deterministic truthful mechanism is known to obtain an approximation ratio better than $O(\frac{m}{\sqrt{\log m}})$ for the general problem [51]. This is true even for the special case where each bidder is interested only in sets of size at most some constant $s \geq 2$ (the s -CA problem), where the standard greedy algorithm obtains an $s + 1$ approximation. The size of this gap between the power of truthful and non-truthful algorithms is discouraging, but makes for a tantalizing open problem.

In the previous two chapters, we have demonstrated that the simple greedy algorithms described above can be paired with straightforward payment schemes to obtain mechanisms with matching performance at equilibrium, for a variety of equilibrium concepts. The simplicity of these mechanisms ignites a hope that perhaps some alternative greedy

The results in this chapter are based upon joint work with Allan Borodin [16].

approaches could generate truthful mechanisms with approximation factors approaching the performance of the standard algorithms. We are therefore motivated to ask the following loosely-defined question: can any “natural” auction that proceeds by ranking bids in some manner, and allocating to the agents with the “best” bids, be simultaneously truthful and achieve a good approximation to the social welfare?

We would argue that the answer to this question is not immediately clear. Indeed, many different auction methods may fit the above description. For instance, a truthful auction due to Bartal et al. [11] for the multi-unit combinatorial auction problem is a primal-dual algorithm that proceeds by iteratively constructing a price vector, and can be viewed as resolving bids in a (specially-tailored and adaptive) greedy manner. This approach is particularly appealing from a practical standpoint, as it mirrors ascending price vector methods currently in practical use. Is it possible that such methodologies, extended further, might lead to a similar greedy-like truthful algorithm for the more general combinatorial auction problem? In this chapter we show that the answer is a resounding *no*.

Our goal is to develop lower bounds for truthful CA mechanisms that satisfy our notion of a “natural” auction alluded to above. We ask: can any truthful *greedy* algorithm obtain an approximation ratio better than $O(\frac{m}{\sqrt{\log(m)}})$? Our specific interest in greedy algorithms is motivated threefold (in addition to symmetry with previous chapters). First, most known examples of truthful, non-MIR algorithms for combinatorial auction problems apply greedy methods [6, 11, 18, 59, 63, 64, 71]; indeed, greedy algorithms embody the conceptual monotonicity properties generally associated with truthfulness, and are thus natural candidates for truthful mechanism construction. Second, greedy algorithms are known to obtain asymptotically tight approximation bounds for many CA problems despite their simplicity. Finally, and perhaps most importantly, many auctions used in practice apply greedy methods, despite the fact that they may not be incentive compatible (e.g. the generalized second price auction for adwords [38]).

That is, simple mechanisms (and in particular greedy mechanisms) seem to be good candidates for auctions due to other considerations beyond truthfulness, such as ease of public understanding and perceived fairness.

We formalize the notion of “greedy algorithm” as the class of *priority algorithms* [17]. This class will prove to be more general than the greedy framework we described previously, and hence the algorithms used to develop mechanisms in the previous two chapters, but this serves to strengthen our (negative) results. Informally speaking, a priority algorithm proceeds by ranking bids according to some (possibly adaptive) quality score; winning a certain bundle then requires that one’s bid be superior (in terms of the ranking) to the conflicting bids with which it competes. Priority algorithms include, for example, many well-known primal-dual algorithms, as well as other greedy algorithms with adaptive and non-trivial selection rules. Moreover, this class is *independent of computational constraints* and also independent of the manner in which valuation functions are accessed. In particular, our results apply to algorithms in the demand query model and the general query model, as well as to auctions in which bids are explicitly represented. Roughly speaking, a priority algorithm has some notion of what constitutes the “best” bid in any given auction instance; the auction finds this bid, satisfies it, then iteratively resolves the reduced auction problem with fewer objects (possibly with an adaptive notion of the “best” bid). For example, the previously mentioned truthful algorithm for multi-unit auctions due to Bartal et al. [11] that updates a price vector while iteratively satisfying agent demands falls into this framework.

Results Our main result demonstrates that if a truthful auction for an s -CA proceeds in the greedy priority-based manner described above, then it cannot perform much better than the naive algorithm that allocates all objects to a single bidder. The gap described in our result is extreme: for $s = 2$, the standard (but non-truthful) greedy algorithm is a 3-approximation for the s -CA problem, but no truthful greedy algorithm can obtain a

sublinear approximation bound.

We then extend our results to related models of priority algorithms. We first consider a model in which the algorithm elicits a collection of bids for desired sets from each agent, and must then proceed using only those bids that were given (and, in particular, cannot allocate a set that was not explicitly bid upon). We show that under this *elementary bids* model, no truthful priority algorithm for the CA problem can obtain a sublinear approximation bound. Note that we have dropped the greediness assumption; this corresponds to allowing the algorithm to decide (irrevocably) to not allocate the highest-priority bid.

Next, we consider a model in which the algorithm views an agent's entire valuation function in a single step. That is, the agents themselves are the items, and the priority algorithm defines a (possibly adaptive) ranking function over the space of all valuations. The priority algorithm must then irrevocably decide which set to allocate to a player when that player is considered. The standard greedy algorithms for the CA and s-CA problems also fall within this player-as-item model. For this model, we again show that no truthful priority algorithm for the CA problem can obtain a sublinear approximation bound.

7.1 Preliminaries

7.1.1 Critical Prices

Recall from Theorem 2.6.1 that a truthful mechanism for a combinatorial auction must charge critical prices as its payments, and must always allocate to each agent a set that maximizes his utility subject to these prices.

Note that the critical price for set S given declaration \mathbf{d}_{-i} , $\theta_i(S, \mathbf{d}_{-i})$ need not be finite. If $\theta_i(S, \mathbf{d}_{-i}) = \infty$, then a mechanism will simply not allocate S to bidder i for any reported valuation d_i . In addition, as we now show, one can assume without loss of generality that critical prices are monotone in the allocated set S .

Claim 7.1.1 *Suppose that a mechanism satisfies the conditions of Theorem 2.6.1 (i.e. the mechanism is truthful). Then one can assume without loss of generality that for all $i \in N$, all $\mathbf{d}_{-i} \in V_{-i}$, and all $S \subseteq T \subseteq M$, $\theta_i(S, \mathbf{d}_{-i}) \leq \theta_i(T, \mathbf{d}_{-i})$.*

Proof: Suppose \mathcal{M} is an incentive compatible mechanism with allocation rule $\mathbf{x}(\cdot)$. Then \mathcal{M} satisfies the critical pricing property, say with prices θ_i . We will construct a new set of critical prices θ_i' that satisfies the conditions of Theorem 2.6.1 while also satisfying the monotonicity conditions of our claim.

For all $i \in N$, $\mathbf{d}_{-i} \in V_{-i}$, and $S \subseteq M$, define $\theta_i'(S, \mathbf{d}_{-i})$ by

$$\theta_i'(S, \mathbf{d}_{-i}) = \min\{\theta_i(T, \mathbf{d}_{-i}) \mid T \supseteq S\}.$$

Notice that $\theta_i'(S, \mathbf{d}_{-i}) \leq \theta_i'(T, \mathbf{d}_{-i})$ whenever $S \subseteq T$. Furthermore, $\theta_i'(S, \mathbf{d}_{-i}) \leq \theta_i(S, \mathbf{d}_{-i})$ for all $S \subseteq M$.

We claim that θ_i' satisfies the conditions of Theorem 2.6.1 for the mechanism \mathcal{M} . Choose any $i \in N$ and $\mathbf{d} \in V$ and suppose that $x_i(\mathbf{d}) = \tilde{S}$. Then, by the critical pricing property (for prices θ_i), $d_i(\tilde{S}, \mathbf{d}_{-i}) - \theta_i(\tilde{S}, \mathbf{d}_{-i}) \geq d_i(T, \mathbf{d}_{-i}) - \theta_i(T, \mathbf{d}_{-i})$ for all $T \subseteq M$, and furthermore $p_i(\mathbf{d}) = \theta_i(\tilde{S}, \mathbf{d})$. We need to show that if the mechanism sets $p_i(\mathbf{d}) = \theta_i'(\tilde{S}, \mathbf{d})$ then $d_i(\tilde{S}, \mathbf{d}_{-i}) - \theta_i'(\tilde{S}, \mathbf{d}_{-i}) \geq d_i(T, \mathbf{d}_{-i}) - \theta_i'(T, \mathbf{d}_{-i})$ for all $T \subseteq M$. We derive the following for all $T \subseteq M$:

$$\begin{aligned} d_i(T, \mathbf{d}_{-i}) - \theta_i'(T, \mathbf{d}_{-i}) &= d_i(T, \mathbf{d}_{-i}) - \theta_i(T', \mathbf{d}_{-i}) \text{ for some } T' \supseteq T \\ &\leq d_i(T', \mathbf{d}_{-i}) - \theta_i(T', \mathbf{d}_{-i}) \text{ by monotonicity of declarations} \\ &\leq d_i(\tilde{S}, \mathbf{d}_{-i}) - \theta_i(\tilde{S}, \mathbf{d}_{-i}) \text{ since the } \theta_i \text{ are critical prices} \\ &\leq d_i(\tilde{S}, \mathbf{d}_{-i}) - \theta_i'(\tilde{S}, \mathbf{d}_{-i}) \text{ by the definition of } \theta_i' \end{aligned}$$

We therefore have that the θ_i' are critical prices for the mechanism, as required.

□

7.1.2 Priority Algorithms

In this section we review the priority algorithm framework [17] and discuss how it can be applied to the CA problem. This will broaden the notion of a greedy algorithm from previous chapters. We view an *input instance* to an algorithm as a subset of *input items* from a known input space \mathcal{I} . Note that \mathcal{I} depends on the problem being considered, and is the set of *all possible* input items: an input instance is a finite subset I of \mathcal{I} . The problem definition may place restrictions on the input: an input instance $I \subseteq \mathcal{I}$ is *valid* if it satisfies all such restrictions. For example, in the CA problem, we would not allow having an agent who values a subset $S' \subset S$ more than the set S . The output of the algorithm is a decision made for each input item in the input instance. For example, these decisions may be of the form “accept/reject”, allocate set S to agent i , etc. The problem may place restrictions on the nature of the decisions made by the algorithm; we say that the output of the algorithm is *valid* if it satisfies all such restrictions. A *priority algorithm* is then any algorithm of the following form:

ADAPTIVE PRIORITY

Input: A set I of items, $I \subseteq \mathcal{I}$

while not empty(I)

Ordering: Choose, without looking at I , a total ordering \mathcal{T} over \mathcal{I}

$next \leftarrow$ first item in I according to ordering \mathcal{T}

Decision: make an irrevocable decision for item $next$

 remove $next$ from I ; remove from \mathcal{I} any items preceding $next$ in \mathcal{T}

end while

We emphasize the importance of the ordering step in this framework: an adaptive priority algorithm is free to choose *any* ordering over the space of possible input items, and can change this ordering adaptively after each input item is considered. Once an item is processed, the algorithm is not permitted to modify its decision. On each iteration a

priority algorithm learns what (higher-priority) items are *not* in the input. A special case of (adaptive) priority algorithms are *fixed order* priority algorithms in which one fixed ordering is chosen before the while loop (i.e. the “ordering” and “while” statements are interchanged). Our inapproximation results for truthful CAs will hold for the more general class of adaptive priority algorithms although many greedy CA algorithms are fixed order.

Admittedly, the term “greedy” implies a more opportunistic aspect than is apparent in the definition of priority algorithms. Indeed, we view priority algorithms more generally as “greedy-like” or “myopic”. A *greedy* priority algorithm satisfies an additional property: the choice made for each input item must optimize the objective of the algorithm as though that item were the last item in the input. We note that many greedy CA algorithms are fixed order greedy priority algorithms.

7.2 Truthful Priority Algorithms

As noted above, Lehmann, O’Callahan and Shoham [64] show that a greedy $O(\sqrt{m})$ -approximation algorithm¹ for combinatorial auctions can be made truthful (using critical pricing) for single-minded bidders, but is not incentive compatible for the more general CA problem. Our high-level goal is to prove that this is a general phenomenon common to all priority algorithms. In order to apply the concept of priority algorithms we must define the set \mathcal{I} of possible input items and the nature of decisions to be made. We consider two natural input formulations: sets as items, and bidders as items. We assume that n , the number of bidders, and m , the number of objects, are known to the mechanism and let $k = \min\{m, n\}$.

¹The Lehmann et al algorithm will satisfy all models discussed in section 7.2.

7.2.1 Sets as Items

In our primary model, we view an input instance to the combinatorial auction problem as a list of set-value pairs for each bidder. Note that this is the notion of a greedy algorithm described in Chapter 2. An item is a tuple (i, S, t) , $i \in N$, $S \subseteq M$, and $t \in \mathbb{R}_{\geq 0}$. A valid input instance $I \subset \mathcal{I}$ contains at most one tuple $(i, S, v_i(S))$ for each $i \in N$ and $S \subseteq M$ and for every pair of tuples (i, S, v) and (i', S', v') in I such that $i = i'$ and $S \subseteq S'$, it must be that $v \leq v'$. We note that since a valid input instance may contain an exponential number of items, this model applies most directly to algorithms that use oracles to query input valuations, such as demand oracles², but it can also apply to succinctly represented valuation functions.³

The decision to be made for item (i, S, t) is whether or not the objects in S should be added to any objects already allocated to bidder i . For example, an algorithm may consider item (i, S_1, t_1) and decide to allocate S_1 to bidder i , then later consider another item (i, S_2, t_2) (where S_2 and S_1 are not necessarily disjoint) and, if feasible, decide to change bidder i 's allocation to $S_1 \cup S_2$.

A *greedy algorithm* in the sets as items model must accept any feasible, profitable item (i, S, t) it considers⁴ Our main result is a lower bound on the approximation ratio achievable by a truthful greedy algorithm in the sets as items model. Theorem 7.2.1 implies a severe separation between the power of greedy algorithms and the power of truthful greedy algorithms. A simple greedy algorithm obtains a 3-approximation for

²It is tempting to assume that this model is equivalent to a value query model, where the mechanism queries bidders for their values for given sets. The priority algorithm model is actually more general, as the mechanism is free to choose an arbitrary ordering over the space of possible set/value combinations. In particular, the mechanism could order the set/value pairs by the utility they would generate under a given set of additive prices, simulating a demand query oracle.

³That is, by assigning priority only to those tuples appearing in a given representation.

⁴That is, when considering a bid (i, S, t) , a greedy algorithm must allocate S to agent i if no objects in S have already been allocated to another bidder, and $d_i(S_1 \cup S) > d_i(S_1)$. In our proof of Theorem 7.2.1, it will always be the case that $S_1 = \emptyset$ (i.e. no items have already been allocated to agent i), so that the greedy assumption is simplified as follows: when considering a bid (i, S, t) , a greedy algorithm must allocate S to agent i if $t > 0$ and no objects in S have already been allocated to another bidder.

the 2-CA problem, yet no truthful greedy priority algorithm (indeed, any algorithm that irrevocably satisfies bids based on a notion of priority) can obtain even a sublinear approximation.

Theorem 7.2.1 *Suppose \mathcal{A} is an incentive compatible greedy priority algorithm that uses sets as items. Then \mathcal{A} cannot approximate the optimal social welfare by a factor of $\frac{(1-\delta)k}{2}$ for any $\delta > 0$. This result also applies to the special case of the 2-CA problem, in which each desired set has size at most 2.*

Before beginning the proof, consider the following intuition as to why such an algorithm \mathcal{A} cannot exist. Suppose some bidder i has a very large value for each of two singletons. Our algorithm \mathcal{A} would surely want to allocate one of these singletons to this bidder. Since \mathcal{A} is greedy, it must do so without first considering the (smaller) values held by other bidders for sets containing those singletons. However, if \mathcal{A} is truthful, then by Theorem 2.6.1 it must also maximize utility for agent i . The algorithm must therefore allocate the singleton which has the smaller critical price. This implies that the relationship between the prices for these singletons must be independent of their value to other bidders! This allows us to show that algorithm \mathcal{A} must have poor performance, since a singleton desired at a high value by many players must have a higher price than a singleton not desired by any other players, in order to guarantee a good approximation ratio.

Proof: Choose $\delta > 0$ and suppose \mathcal{A} obtains a bounded approximation ratio. For each $i \in N$, let V_{-i}^+ be the set of valuations with the property that $v_\ell(S) > 0$ for all $\ell \neq i$ and all non-empty $S \subseteq M$. The heart of our proof is the following claim, which shows that the relationship between critical prices for singletons for one bidder is independent of the valuations of other bidders. Recall that $\theta_i(S, \mathbf{d}_{-i})$ is the critical price for set S for bidder i , given \mathbf{d}_{-i} .

Lemma 7.2.2 *For all $i \in N$, and for all $a, b \in M$, either $\theta_i(\{a\}, \mathbf{d}_{-i}) \geq \theta_i(\{b\}, \mathbf{d}_{-i})$ for all $\mathbf{d}_{-i} \in V_{-i}^+$, or $\theta_i(\{a\}, \mathbf{d}_{-i}) \leq \theta_i(\{b\}, \mathbf{d}_{-i})$ for all $\mathbf{d}_{-i} \in V_{-i}^+$. This is true even when agents desire sets of size at most 2.*

Proof: Choose $i \in N$, $a, b \in M$, and $\mathbf{d}_{-i}, \mathbf{d}_{-i}' \in V_{-i}^+$. Suppose for contradiction that $\theta_i(\{a\}, \mathbf{d}_{-i}) > \theta_i(\{b\}, \mathbf{d}_{-i})$ but $\theta_i(\{b\}, \mathbf{d}_{-i}') > \theta_i(\{a\}, \mathbf{d}_{-i}')$. We will consider a number of possible valuations to be declared by our bidders.

Let v^* be the maximum value assigned to any set by any player in \mathbf{d}_{-i} or \mathbf{d}_{-i}' . Then note that the maximum social welfare that can be obtained is $(k-1)v^*$ if bidder i does not participate and other bidders declare values \mathbf{d}_{-i} or \mathbf{d}_{-i}' . Let $x = k^2v^*$. We will define various different possible valuation functions for bidder i : f , h , and g_c for all $c \in M$.

$$f(S) = \begin{cases} x & \text{if } a \in S \\ x & \text{if } b \in S \\ 0 & \text{otherwise.} \end{cases} \quad g_c(S) = \begin{cases} \epsilon & \text{if } a \in S, c \notin S \\ \epsilon & \text{if } b \in S \\ x & \text{if } \{a, c\} \subseteq S \\ 0 & \text{otherwise.} \end{cases}$$

$$h(S) = \begin{cases} \epsilon & \text{if } a \in S \\ \epsilon & \text{if } b \in S \\ 0 & \text{otherwise.} \end{cases}$$

Note that each of these valuation profiles can be interpreted as a profile in which the agent desires sets of size at most 2. Note also that g_a and g_b are well-defined: the former assigns value x to any set containing \mathcal{A} , and the latter assigns value x to any set containing both \mathcal{A} and b .

We are now ready to discuss the behaviour of algorithm \mathcal{A} . Consider the subset $\mathcal{I}_1 \subset \mathcal{I}$ that contains the following input items: $(i, S, f(S))$ and $(i, S, h(S))$ for every $S \subseteq M$; $(i, S, g_c(S))$ for all $c \in M$ and $S \subseteq M$; and $(j, S, d_j(S))$, $(j, S, d_j'(S))$, (j, S, ϵ) , and (j, S, v^*) for all $j \neq i$ and $S \subseteq M$. In other words, \mathcal{I}_1 contains all of the input items

consistent with the valuation functions we defined above, plus input items (j, S, ϵ) and (j, S, v^*) for each set S and each bidder $j \neq i$.

We know that if \mathcal{A} is a priority algorithm, then it must have some initial ordering over \mathcal{I} , and hence over \mathcal{I}_1 . Consider the first item in \mathcal{I}_1 under this ordering. We consider different cases for the nature of this item.

Case 1: (j, S, t) , $j \neq i$. Then $t \in \{d_j(S), d_j'(S), \epsilon, v^*\}$ and hence $t > 0$. Choose any $c \in S$. Let I_1 be a valid input instance consisting of items from \mathcal{I}_1 , such that $(j, S, t) \in I_1$ and I_1 is consistent with agent i having valuation g_c . Note that such an I_1 always exists; for example, if $t = d_j(S)$ we could set I_1 to be consistent with each agent $\ell \neq i$ having valuation d_ℓ . Then $I_1 \subseteq \mathcal{I}_1$ and $(j, S, t) \in I_1$, so item (j, S, t) will be considered first by algorithm \mathcal{A} on input I_1 .

Since \mathcal{A} is greedy, \mathcal{A} will allocate set S to bidder j . Then it must be that, in the final allocation, bidder i is not allocated any set containing c . Thus, from the definition of g_c , bidder i obtains a value of at most ϵ . Furthermore, all other bidders can obtain a total welfare of at most $(k-1)v^*$, for a total social welfare of at most $(k-1)v^* + \epsilon$. On the other hand, a total of at least $x = k^2v^*$ is possible by allocating $\{a, c\}$ to bidder i . Then as long as $\epsilon < v^*$ the approximation ratio obtained by \mathcal{A} is at least k , a contradiction.

Case 2: (i, S, x) , $a \in S$ or $b \in S$. By symmetry we can assume $a \in S$. Consider the input instance I_2 in which bidder i declares valuation f , and every other bidder $j \neq i$ declares valuation d_j . Then $f(S) = x$, so $(i, S, x) \in I_2 \subseteq \mathcal{I}_1$, and therefore \mathcal{A} will consider item (i, S, x) first on input I_2 . Since $x > 0$ and \mathcal{A} is greedy, the algorithm will assign set S to bidder i .

Suppose that in the final allocation, bidder i is allocated some set $T \supseteq S$. Then since $a \in T$, we know that $\theta_i(T, \mathbf{d}_{-i}) \geq \theta_i(\{a\}, \mathbf{d}_{-i}) > \theta_i(\{b\}, \mathbf{d}_{-i})$. But note $f(T) = f(\{a\}) = x$, so that $f(T) - \theta_i(T, \mathbf{d}_{-i}) < f(\{b\}) - \theta_i(\{b\}, \mathbf{d}_{-i})$. In other words, \mathcal{A} does not maximize the utility of player i . By Theorem 2.6.1, \mathcal{A} is not incentive compatible, a contradiction.

Case 3: (i, S, ϵ) , $a \in S$ or $b \in S$. By symmetry we can assume $a \in S$. Consider the

input instance I_3 in which bidder i declares valuation h , and every other bidder $j \neq i$ declares valuation d_j . Then $(i, S, \epsilon) \in I_3 \subseteq \mathcal{I}_1$, so \mathcal{A} will consider item (i, S, ϵ) first on input I_3 . From this point, we obtain a contradiction in precisely the same way as in Case 2.

Case 4: (i, S, t) , $a \notin S$ and $b \notin S$. Then from the definitions of f , g_c , and h , we must have $t = 0$. Thus when processing this item, \mathcal{A} is free to allocate S to bidder i or not. If \mathcal{A} does not allocate S to i , then we will consider the *next* item considered by the algorithm \mathcal{A} , and repeat our case analysis. The case analysis proceeds in the same way, since no objects would have been allocated. This process must terminate, as algorithm \mathcal{A} must eventually consider some set S for agent i that contains either a or b , or (reasoning as above) some set for agent $j \neq i$.

Suppose, on the other hand, that \mathcal{A} does allocate S to i . Then consider the input instance I_4 in which bidder i declares valuation h and all other bidders declare the following valuation f_S :

$$f_S(T) = \begin{cases} v^* & \text{if } S \subseteq T \\ \epsilon & \text{otherwise.} \end{cases}$$

We note that valuation f_S defines the value of any set to be either ϵ or v^* , so in particular $I_3 \subseteq \mathcal{I}_1$. Since $(i, S, 0) \in I_3$, this item will be considered first by \mathcal{A} on input I_3 , and S will be allocated to player i . But then in the final allocation each other bidder can obtain a welfare of at most ϵ , for a total welfare of at most $k\epsilon$. On the other hand, a welfare of v^* was possible by allocating S to any bidder other than bidder i . Thus, if we choose $\epsilon < v^*/k^2$ we conclude that \mathcal{A} has an approximation ratio of at least k , a contradiction.

We have shown that every case leads to a contradiction, completing the proof of Lemma 7.2.2. \square

We can think of Lemma 7.2.2 as defining, for each $i \in N$, an ordering over the elements of M . For each $i \in N$ and $a, b \in M$, write $a \preceq_i b$ to mean $\theta_i(a, \mathbf{d}_{-i}) \leq \theta_i(b, \mathbf{d}_{-i})$ for all $\mathbf{d}_{-i} \in V_{-1}^+$. For all $i \in N$ and $a \in M$, define $T_i(a) = \{a_j : a \preceq_i a_j\}$. That is, $T_i(a)$

is the set of objects that have higher price than \mathcal{A} for agent i . Our next claim shows a strong relationship between whether \mathcal{A} is allocated to bidder i and whether any object in $T_i(a)$ is allocated to bidder i .

Lemma 7.2.3 *Choose $a \in M$, $i \in N$, and $S \subseteq M$, and suppose $S \cap T_i(a) \neq \emptyset$. Choose some $d_i \in V_i$ and suppose that $d_i(\{a\}) > d_i(S)$. Then if $\mathbf{d}_{-i} \in V_{-i}^+$, bidder i cannot be allocated set S by algorithm \mathcal{A} given input \mathbf{d} .*

Proof: We know that $\theta_i(S, \mathbf{d}_{-i}) \geq \theta_i(\{a_j\}, \mathbf{d}_{-i})$ for any $a_j \in S$. Thus, regardless of the choice of \mathbf{d}_{-i} ,

$$\theta_i(S, \mathbf{d}_{-i}) \geq \max_{a_j \in S \cap T_i(a)} (\theta_i(\{a_j\}, \mathbf{d}_{-i})) \geq \theta_i(\{a\}, \mathbf{d}_{-i})$$

from the definition of $T_i(a)$. Since $d_i(a) > d_i(S)$, this implies that $d_i(a) - \theta_i(\{a\}, \mathbf{d}_{-i}) > d_i(S) - \theta_i(S, \mathbf{d}_{-i})$, so by Theorem 2.6.1 bidder i cannot be allocated set S , as required.

□

Lemma 7.2.3 is strongest when $T_i(a)$ is large; that is, when \mathcal{A} is “small” in the ordering \preceq_i . We therefore wish to find an object of M that is small according to many of these orderings, simultaneously. Let $R(a) = \{i \in N : |T_i(a)| \geq k/2\}$, so $R(a)$ is the set of players for which there are at least $k/2$ objects greater than \mathcal{A} . The next claim follows by a straightforward counting argument.

Lemma 7.2.4 *There exists $a^* \in M$ such that $|R(a^*)| \geq k/2$.*

Proof: We note that

$$\sum_{i \in N} \sum_{\substack{a \in M \\ |T_i(a)| \geq k/2}} 1 = \sum_{i \in N} (m - k/2) = n(m - k/2).$$

Rearranging order of summation, we also have

$$\sum_{i \in N} \sum_{\substack{a \in M \\ |T_i(a)| \geq k/2}} 1 = \sum_{a \in M} \sum_{\substack{i \in N \\ |T_i(a)| \geq k/2}} 1 = \sum_{a \in M} |S(a)|.$$

We conclude that $\sum_{a \in M} |S(a)| = n(m - k/2)$, so there must exist some $a^* \in M$ such that $|S(a^*)| \geq \frac{n(m-k/2)}{m}$. We know that either $n \geq m = k$ or $m \geq n = k$; in either case we obtain $|S(a^*)| \geq \frac{n(m-k/2)}{m} \geq k/2$ as required. \square

We are now ready to proceed with the proof of Theorem 7.2.1. Let $a^* \in M$ be the object from Lemma 7.2.4. Let $\epsilon > 0$ be a sufficiently small value to be defined later. We now define a particular input instance to algorithm \mathcal{A} . For each $i \in R(a^*)$, bidder i will declare the following valuation function, d_i :

$$d_i(S) = \begin{cases} 1 & \text{if } a^* \in S \\ 1 - \delta/2 & \text{if } a^* \notin S \text{ and } S \cap (T_i(a^*)) \neq \emptyset \\ \epsilon & \text{otherwise.} \end{cases}$$

Each bidder $i \notin R(a^*)$ will declare a value of ϵ for every set.

For each $i \in R(a^*)$, $d_i(a_j) \geq 1 - \delta/2$ for every $a_j \in T_i(a^*)$. Since $|R(a^*)| \geq k/2$ and $|T_i(a^*)| \geq k/2$, it is possible to obtain a social welfare of at least $\frac{(1-\delta/2)k}{2}$ by allocating singletons to bidders in $R(a^*)$.

Consider the social welfare obtained by algorithm \mathcal{A} . The algorithm can allocate object a^* to at most one bidder, say bidder i , who will obtain a social welfare of at most 1. For any bidder $\ell \in R(a^*)$, $\ell \neq i$, $d_\ell(S) = 1 - \delta/2 < 1$ for any S containing elements of $T_\ell(a^*)$ but not a^* . Thus, by Lemma 7.2.3, no bidder in $R(a^*)$ can be allocated any set S that contains an element of $T_i(a^*)$ but not a^* . Therefore every bidder other than bidder i can obtain a value of at most ϵ , for a total social welfare of at most $1 + k\epsilon$.

We conclude that algorithm \mathcal{A} has an approximation factor no better than $\frac{k(1-\delta/2)}{2(1+k\epsilon)}$. Choosing $\epsilon < \frac{\delta}{2(1-\delta)k}$ results in a bound of at least $\frac{k(1-\delta)}{2}$, completing the proof of Theorem 7.2.1. \square

We believe that the greediness assumption of Theorem 7.2.1 can be removed. As partial progress toward this goal, we show that this assumption can be removed if we restrict our attention to the following alternative input model for priority algorithms,

in which an algorithm can only consider and allocate sets whose values are explicitly represented (i.e. not implied by the value of a subset).

7.2.2 Elementary bids as items

Consider an auction setting in which agents do not provide entire valuation functions, but rather each agent specifies a list of *desired sets* S_1, \dots, S_k and a value for each one. Moreover, each agent receives either a desired set or the empty set. This can be thought of as an auction with a succinct representation for valuation functions, in the spirit of the XOR bidding language [74]. We model such an auction as a priority algorithm by considering items to be the bids for desired sets. In such a setting, the specified set-value pairs are called *elementary bids*. We say that the priority model uses *elementary bids as items* when only elementary bids $(i, S, v(S))$ can be considered by the algorithm. For each item $(i, S, v(S))$, the decision to be made is whether or not S will be the one and only one set allocated to agent i ; that is, whether or not the elementary bid for S will be “satisfied.” In particular, unlike in the sets as items model, we do not permit the algorithm to build up an allocation incrementally by accepting many elementary bids from a single agent.

We now show that the greediness assumption from Theorem 7.2.1 can be removed when we consider priority algorithms in the elementary bids as items model.

Theorem 7.2.5 *Suppose \mathcal{A} is an incentive compatible priority algorithm for the CA problem that uses elementary bids as items. Then \mathcal{A} cannot approximate the optimal social welfare by a factor of $(1 - \delta)k$ for any $\delta > 0$.*

Proof: We first note that if only bidder i places bids, then $\theta_i(M, \mathbf{d}_{-i}) = 0$.

Let I_1 be an input instance containing items $(i, M, 1 + \delta)$ and $(i, S, 1)$ for all $S \neq M$, for each $1 \leq i \leq N$. That is, each bidder has a value of 1 for each singleton and $1 + \delta$ for the set of all objects. Then \mathcal{A} must consider some input item first given input I_1 ; suppose

the first item has corresponding bidder j . Now consider cases based on the nature of the first item.

Case 1: $(j, M, 1 + \delta)$. Consider the decision made by \mathcal{A} for this item. If \mathcal{A} allocates M to j , then for input instance I_1 \mathcal{A} obtains a social welfare of $1 + \delta$, whereas the optimal welfare is k . Thus \mathcal{A} has an approximation ratio no better than $(1 + \delta)^{-1}k > (1 - \delta)k$, a contradiction. Next suppose \mathcal{A} does not allocate M to j . Consider input instance $I_2 \subset I_1$ that contains only item $(j, M, 1 + \delta)$. Then \mathcal{A} cannot distinguish between I_1 and I_2 when considering item $(j, M, 1 + \delta)$. Thus \mathcal{A} will not allocate M to bidder j on input I_2 , which contradicts Theorem 2.6.1.

Case 2: $(j, S, 1)$, $S \neq M$. Consider the decision made by \mathcal{A} for this item. Suppose \mathcal{A} does not allocate S to bidder j . Let $I_3 \subseteq I_1$ be the input instance consisting only of items $(j, T, 1)$ for all $T \supseteq S$; that is, player j has a single-minded valuation for set S . Since \mathcal{A} cannot distinguish between I_1 and I_3 when considering item $(j, S, 1)$, it must be that \mathcal{A} does not allocate S to bidder j on input I_3 . Since \mathcal{A} does not allocate any set T to player j other than set S (by assumption), it must not allocate anything to player j . Thus \mathcal{A} obtains a social welfare of 0 when 1 was possible, contradicting the supposed approximation ratio of \mathcal{A} .

Thus \mathcal{A} must allocate S to bidder j on input I_3 . Let $I_4 \subseteq I_1$ be the input instance consisting of items $(j, S, 1)$ and $(j, M, 1 + \epsilon)$. Then \mathcal{A} will allocate S to bidder j in instance I_4 , but this contradicts Theorem 2.6.1 (which requires that \mathcal{A} allocate M to bidder j).

We therefore arrive at a contradiction in all cases, as required. \square

7.2.3 Bidders as Items

Roughly speaking, the lower bounds in Theorems 7.2.1 and 7.2.5 follow from a priority algorithm's inability to determine which of many different mutually-exclusive desires of an agent to consider first when constructing an allocation. One might guess that such

difficulties can be overcome by presenting an algorithm with more information about an agent's valuation function at each step. To this end, we consider an alternative model of priority algorithms in which the agents themselves are the items, and the algorithm is given complete access to an agent's declared valuation function each round.

Under this model, \mathcal{I} consists of all pairs (i, v_i) , where $i \in N$ and $v_i \in V_i$. A valid input instance contains one item for each bidder. The decision to be made for item (i, v_i) is a set $S \subseteq M$ to assign to bidder i . The truthful greedy CA mechanism for single-minded bidders due to Lehmann et al. [64] falls within this model, as does its (non-truthful) generalization to complex bidders [64], the primal-dual algorithm of [18], and the (first) algorithm of [11] for multi-unit CAs. We now establish an inapproximation bound for truthful priority allocations that use bidders as items.

Theorem 7.2.6 *Suppose \mathcal{A} is an incentive compatible priority algorithm for the (2-minded) CA problem that uses bidders as items. Then \mathcal{A} cannot approximate the optimal social welfare by a factor of $\frac{(1-\delta)k}{2}$ for any $\delta > 0$.*

Proof: Choose $\delta > 0$ and suppose for contradiction that \mathcal{A} is an incentive compatible adaptive priority algorithm that achieves an approximation ratio of $k(1 - \delta)/2$. Recall that an item is a tuple (i, v_i) , where $1 \leq i \leq n$ is a bidder and $v_i : 2^M \rightarrow \mathbb{R}$ is a valuation function.

We will construct a set of input instances for which \mathcal{A} is forced to make a particular allocation, due to incentive compatibility. We define two sets of valuation functions, $\{g_1, \dots, g_k\}$ and $\{f_1, \dots, f_k\}$, that will be used in these input instances. The functions g_1, \dots, g_k are straightforward: for each $1 \leq i \leq k$, define valuation function g_i by

$$g_i(S) = \begin{cases} 1 & \text{if } a_i \in S \\ 0 & \text{otherwise.} \end{cases}$$

Then g_i is a single-minded valuation function, where the desired set is $\{a_i\}$ with value 1.

The definition of valuation functions f_1, \dots, f_k is more involved. Fix $i \in N$ and define $V'_{-i} := \{g_1, \dots, g_k\}^{n-1}$. Consider an instance \mathbf{d} of the combinatorial auction problem in which $\mathbf{d}_{-i} \in V'_{-i}$. That is, each bidder $j \neq i$ is single-minded, and desires a singleton with value 1. By the critical price property, there is a critical price $\theta_i(M, \mathbf{d}_{-i})$ for set M given this \mathbf{d}_{-i} .

Lemma 7.2.7 $\theta_i(M, \mathbf{d}_{-i}) \leq kn$.

Proof: Suppose otherwise that $\theta_i(M, \mathbf{d}_{-i}) > kn$. Suppose further that bidder i is single-minded with desired set M , and with $d_i(M) = kn$. Then $d_i(M) - \theta_i(M, \mathbf{d}_{-i}) < 0 = d_i(\emptyset) - \theta_i(\emptyset, \mathbf{d}_{-i})$. Therefore, by the critical pricing property, \mathcal{A} cannot allocate M to bidder i , and hence bidder i obtains a value of 0. Now consider the social welfare obtained by \mathcal{A} : it can be at most $n - 1$, since bidder i obtains a welfare of 0 and each other bidder has value at most 1 for any set. The optimal social welfare is kn , obtained by allocating M to bidder i . Hence \mathcal{A} obtains an approximation ratio of $\frac{kn}{n-1} > \frac{k(1-\delta)}{2}$ for this input instance, which is a contradiction. This completes the proof of Lemma 7.2.7. \square

We are now ready to define the valuations f_1, \dots, f_k . They are based on values $x, y \in \mathbb{R}$. Define $x \in \mathbb{R}$ as follows:

$$x := 1 + \max_{i \in N} \max_{\mathbf{v}_{-i} \in V'_{-i}} \{\theta_i(M, \mathbf{v}_{-i})\}.$$

That is, x is a value greater than the maximum of the critical price for M for bidder i , over all choices of i and possible desires of singletons with value 1 by other bidders. Set $y := x\delta^{-1}$.

For each $1 \leq i \leq k$, define valuation function f_i as

$$f_i(S) = \begin{cases} y & \text{if } \{a_i\} \subseteq S \subset M \\ y + x & \text{if } S = M \\ 0 & \text{otherwise.} \end{cases}$$

Then $f_i(S)$ is a 2-minded valuation function. We now consider the following subset $\mathcal{I}' \subseteq \mathcal{I}$ of possible input items: \mathcal{I}' contains all bidder-valuation pairs of the form (i, v_i) where $1 \leq i \leq n$ and $v_i = f_j$ or $v_i = g_j$ for some $1 \leq j \leq k$. Note that \mathcal{I}' is not a valid input instance; we think of \mathcal{I}' simply as a subset of \mathcal{I} .

The following claim exploits the incentive compatibility of \mathcal{A} .

Lemma 7.2.8 *Suppose $I = \{(1, d_1), \dots, (n, d_n)\}$ is a valid input instance, in which there exists $i \in N$ such that $d_i \in \{f_1, \dots, f_k\}$, and for all $j \neq i$, $d_j \in \{g_1, \dots, g_k\}$. Then on input I , \mathcal{A} must allocate M to bidder i and \emptyset to all other bidders.*

Proof: For this input instance we have that $\mathbf{d}_{-i} \in V'_{-i}$. Then $x > \theta_i(M, \mathbf{v}_{-i})$ from the definition of x . But now, from the definition of f_i ,

$$d_i(M) - \theta_i(M, \mathbf{v}_{-i}) > (y + x) - x = y \geq d_i(S) \geq d_i(S) - \theta_i(S, \mathbf{d}_{-i})$$

for all $S \neq M$. Therefore, by the critical pricing property (Theorem 2.6.1), \mathcal{A} must allocate M to bidder i , completing the proof of Lemma 7.2.8. \square

Our next step is to construct an input instance $I \subseteq \mathcal{I}'$ on which \mathcal{A} obtains a poor approximation ratio. To do this we will rely on the following claim which will be proven by induction on i .

Lemma 7.2.9 *There exists a labelling of bidders and objects such that the following is true for all $0 \leq i < k/2$. Define $I_i := \{(j, g_j) \mid j \leq i\}$. Then for any valid input instance I such that $I_i \subseteq I \subseteq \mathcal{I}'$, \mathcal{A} will consider all the items in I_i before all other items in I , and will choose to assign \emptyset for each of the items in I_i .*

Proof: We proceed by induction on i . The base case holds by taking $I_0 = \emptyset$. For general $i \geq 1$, suppose the claim is true for $i - 1$. Then $I_{i-1} = \{(1, g_1), \dots, (i-1, g_{i-1})\}$. Define $\mathcal{I}_i \subseteq \mathcal{I}'$ as follows:

$$\mathcal{I}_i := \{(j, v_j) \mid (j, v_j) \in \mathcal{I}', j \geq i\}.$$

That is, \mathcal{I}_i contains items of \mathcal{I}' corresponding to bidders that are not present in I_{i-1} . Then note that if I is a valid input instance such that $I_{i-1} \subseteq I$, then $I \subseteq I_{i-1} \cup \mathcal{I}_i$.

Consider the execution of \mathcal{A} on any valid input instance $I \subseteq I_{i-1} \cup \mathcal{I}_i$. The algorithm will first consider the items of I_{i-1} and allocate \emptyset to each bidder $1, \dots, i-1$ (by assumption). Once this is done, the algorithm will choose an ordering \mathcal{T} over \mathcal{I}_i and examine the next item in I according to \mathcal{T} .

Some item $(j, v_j) \in \mathcal{I}_i$ must come first under this ordering \mathcal{T} . Without loss of generality (by relabeling indices) this item is (i, f_i) or (i, g_i) . We consider these two cases separately.

Case 1: The first item is (i, f_i) . In this case we will choose I so that $(i, f_i) \in I$. Then \mathcal{A} must consider this item next when processing input instance I , and \mathcal{A} must assign some set S to bidder i . If $S = M$, then we will choose I to contain (j, f_j) for all $j > i$; note that $I \subseteq I_{i-1} \cup \mathcal{I}_i$ as required. Since \mathcal{A} allocated M to bidder i , it obtains a social welfare of $x + y$ on input I . However, the optimum welfare is at least $(k - i + 1)y$, since this can be attained by allocating $\{a_j\}$ to bidder j for all $i \leq j \leq k$. Thus the approximation ratio obtained by \mathcal{A} is at least

$$\frac{(k - i + 1)y}{x + y} > \frac{(k/2)y}{y(1 + \delta)} > \frac{(1 - \delta)k}{2},$$

a contradiction.

If, on the other hand, $S \neq M$, we choose I to contain (j, g_j) for all $j > i$. Then I satisfies the requirements of Lemma 7.2.8, so \mathcal{A} must allocate M to bidder i . This is a contradiction. We conclude that this first case cannot occur.

Case 2: The item is (i, g_i) . In this case we will choose I so that $(i, g_i) \in I$. As in the previous case, \mathcal{A} must consider this item next in I , and assign some set S to bidder i . Suppose $S \neq \emptyset$. Then we will choose I to contain $(i + 1, f_{i+1})$, and also (j, g_j) for all $j > i + 1$. Note that then $I \in I_{i-1} \cup \mathcal{I}_i$ as required. Also, in this instance of a combinatorial auction, $v_{-(i+1)}$ contains only single-minded valuations for singletons with value 1. Thus, by the same argument used in Case 1, it must be that bidder $i + 1$ is

allocated M . However, this is not possible, since bidder i is assigned $S \neq \emptyset$. This is a contradiction. We conclude that in this case, bidder i must be assigned \emptyset .

This ends our case analysis. We conclude that item (i, g_i) must occur first in \mathcal{I}_{i-1} in ordering \mathcal{T} , and furthermore if $(i, g_i) \in I$ then \mathcal{A} will consider (i, g_i) next after processing the items in I_{i-1} and will assign \emptyset to bidder i . We can therefore set $I_i = I_{i-1} \cup \{(i, g_i)\}$ to satisfy the requirements of the Lemma 7.2.9. \square

Now suppose $I_{k/2-1}$ is the set from Lemma 7.2.9 with $i = k/2 - 1$. Define input instance I by

$$I := I_{k/2-1} \cup \{(j, g_{k/2}) \mid k/2 \leq j \leq k\}.$$

Note that I is a valid input instance and $I_{k/2-1} \subseteq I \subseteq \mathcal{I}'$. Then by Lemma 7.2.9, algorithm \mathcal{A} must assign \emptyset to each of bidders $1, \dots, k/2 - 1$. Therefore \mathcal{A} can obtain a social welfare of at most 1, by assigning $\{a_{k/2}\}$ to some bidder $j \geq k/2$. However, the optimal social welfare is $k/2$, by assigning $\{a_i\}$ to bidder i for all $1 \leq i \leq k/2$. Hence \mathcal{A} obtains an approximation no better than $k/2$, which is a contradiction. This completes the proof of Theorem 7.2.6. \square

Chapter 8

Conclusions

In this work we have examined a number of central algorithmic mechanism design problems through the lens provided by models of rationality with uncertainty, where we think of agents as having partial information about their opponents. In the two parts of the thesis we applied this modelling choice in very different ways. In the first part, we used the partial information setting to overcome traditional difficulties in truthful mechanism design. The result is a very general black-box transformation that converts arbitrary algorithms for single-parameter problems into truthful mechanisms without loss – a feat that we show is not possible when we restrict ourselves to dominant strategy truthful mechanisms in full-information models. In the second part, we used the partial information setting to motivate an equilibrium solution concept that would be unreasonable in settings of complete information. We showed that simple greedy algorithms are appropriate for use as mechanisms under the goal of achieving good performance at equilibrium. Moreover, these desirably simple algorithms cannot be used to create truthful mechanisms in the full-information model.

There is, of course, much work left to be done, and many related research topics remain open. First, in regard to black-box reductions in single-parameter mechanism design, we provided a reduction in Chapter 3 that generates Bayesian incentive compatible

mechanisms with no loss in expected welfare, and we provided a negative result in Chapter 4 reductions that generate dominant strategy truthful mechanisms with no loss in worst-case welfare approximation. However, this leaves open the question of whether it is possible to convert algorithms into Bayesian incentive compatible mechanisms in such a way that worst-case approximation factors are retained. Alternatively, can arbitrary algorithms be converted into ex post incentive compatible mechanisms without loss to their expected performance with respect to a distribution over types? Can our impossibility result be improved to demonstrate that no randomized black-box transformation can guarantee truthfulness in expectation without loss? What if we restrict ourselves to the subclass of downward-closed feasibility problems?

Also, our results in the first part of the thesis apply only to the objective of social welfare maximization. It would be nice to extend our result more generally to any monotone objective function, such as makespan. Is there a polynomial-time reduction that turns any approximation algorithm for any monotone objective into a Bayesian incentive compatible mechanism with the same approximation factor?

More broadly, while we demonstrated in Chapter 4 that no black-box transformation can convert approximation algorithms into deterministic truthful mechanisms, this does not necessarily imply a gap between the power of Bayesian incentive compatible mechanisms and (deterministic) ex post incentive compatible mechanisms. Is there a single-parameter social welfare problem for which every polytime dominant strategy incentive compatible mechanism obtains social welfare much worse than that of the best possible polytime algorithm? Such a gap would separate the power of Bayesian and non-Bayesian truthful mechanisms for single-parameter problems.

For the second part of the thesis, one direction of future research regards extending our results to other classes of problems. Our methods in Chapters 5 and 6 depend on the structure of greedy algorithms for combinatorial allocation problems, but the overall approach may generalize further. Given a mechanism design problem, when can a black-

box algorithm for the underlying optimization problem be converted into a mechanism that obtains (nearly) the same approximation ratio at every Bayes-Nash equilibrium? What if we also require that agents can easily find equilibria or approximate equilibria, or otherwise relax the equilibrium notion so that natural models of agent behaviour are captured? A more direct question left open by Chapter 6 is whether one can obtain an improved mechanism and/or analysis for best-response bidders that overcomes the constant-factor loss in Theorem 6.3.11 and Theorem 6.3.12.

Another direction of future research would be to widen the scope of a systematic search for truthful approximation algorithms for various problems, as begun in Chapter 7. One might consider randomized priority algorithms [2], priority algorithms with revocable acceptances [52], priority stack algorithms [15, 10], or any number of other algorithm classes. The goal of such an endeavour would be to find mechanisms that are not only truthful, but also satisfy some notion of “naturalness” that is desirable in practical auction implementations.

Our overall conclusion is that while full-information models of agent rationality currently dominate the algorithmic mechanism design literature, a relaxation to partial-information models is well-motivated and provides additional power in solving central problems in the field. We submit that this Bayesian perspective is a useful tool that can be applied to overcome apparent difficulties faced when designing algorithmic ex post incentive compatible mechanisms.

Bibliography

- [1] G. Aggarwal and J. Hartline. Knapsack auctions. In *Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms*, 2006.
- [2] S. Angelopoulos and A. Borodin. Randomized priority algorithms. *Theoretical Computer Science*, 411:2542–2558, 2010.
- [3] A. Archer, C. Papadimitriou, K. Talwar, and E. Tardos. An approximate truthful mechanism for combinatorial auctions with single parameter agents. In *Proc. 14th ACM Symp. on Discrete Algorithms*. ACM/SIAM, 2003.
- [4] A. Archer and E. Tardos. Truthful mechanisms for one-parameter agents. In *Proc. 42nd IEEE Symp. on Foundations of Computer Science*, 2001.
- [5] L. M. Ausubel and P.R. Milgrom. Ascending auctions with package bidding. *Frontiers of Theoretical Economics*, 1(1), 2002.
- [6] Y. Azar, I. Gamzu, and S. Gutner. Truthful unsplittable flow for large capacity networks. In *Proc. 19th ACM Symp. on Parallel Algorithms and Architectures*, 2007.
- [7] M. Babaioff and L. Blumrosen. Computationally-feasible truthful auctions for convex bundles. In *Proc. 7th Intl. Workshop on Approximation Algorithms for Combinatorial Optimization Problems*, 2004.

- [8] M. Babaioff, R. Lavi, and E. Pavlov. Single-value combinatorial auctions and algorithmic implementation in undominated strategies. *Journal of the ACM*, 2009.
- [9] P. Baptiste. Polynomial time algorithms for minimizing the weighted number of late jobs on a single machine with equal processing times. *Journal of Scheduling*, 2:245–252, 1999.
- [10] A. Bar-Noy, R. Bar-Yehuda, A. Freund, J. Naor, and B. Schieber. A unified approach to approximating resource allocation and scheduling. *Journal of the ACM*, 48:1069–1090, 2001.
- [11] Y. Bartal, R. Gonen, and N. Nisan. Incentive compatible multi unit combinatorial auctions. In *Proc. 9th Conf. on Theoretical Aspects of Rationality and Knowledge*, 2003.
- [12] X. Bei and Z. Huang. Bayesian incentive compatibility via fractional assignments. In *Proc. 22nd ACM Symp. on Discrete Algorithms*, 2011.
- [13] K. Bhawalkar and T. Roughgarden. Welfare guarantees for combinatorial auctions with item bidding. In *Proc. 22nd ACM Symp. on Discrete Algorithms*, 2011.
- [14] A. Blum, M. Hajiaghayi, K. Ligett, and A. Roth. Regret minimization and the price of total anarchy. In *Proc. 39th ACM Symp. on Theory of Computing*, 2008.
- [15] A. Borodin, D. Cashman, and A. Magen. How well can primal-dual and local-ratio algorithms perform? In *Proc. 32nd Intl. Colloq. on Automata, Languages and Programming*, 2005.
- [16] A. Borodin and B. Lucier. Greedy mechanism design for truthful combinatorial auctions. In *Proc. 37th Intl. Colloq. on Automata, Languages and Programming*, 2010.

- [17] A. Borodin, M. N. Nielsen, and C. Rackoff. (incremental) priority algorithms. In *Proc. 13th ACM Symp. on Discrete Algorithms*, 2002.
- [18] P. Briest, P. Krysta, and B. Vöcking. Approximation techniques for utilitarian mechanism design. In *Proc. 36th ACM Symp. on Theory of Computing*, 2005.
- [19] D. Buchfuhrer, S. Dughmi, H. Fu, R. Kleinberg, E. Mossel, C. Papadimitriou, M. Schapira, Y. Singer, and C. Umans. Inapproximability for vcg-based combinatorial auctions. In *Proc. 21st ACM Symp. on Discrete Algorithms*, 2010.
- [20] I. Caragiannis, P. Kanellopoulos, C. Kaklamanis, and M. Kyropoulou. On the efficiency of equilibria in generalized second price auctions. In *Proc. 13th ACM Conf. on Electronic Commerce*, 2011.
- [21] C. Chekuri and I. Gamzu. Truthful mechanisms via greedy iterative packing. In *Proc. 12th Intl. Workshop on Approximation Algorithms for Combinatorial Optimization Problems*, 2009.
- [22] G. Christodoulou and A. Kovacs. A deterministic truthful ptas for scheduling related machines. In *Proc. 21st ACM Symp. on Discrete Algorithms*, 2010.
- [23] G. Christodoulou, A. Kovács, and Michael Schapira. Bayesian combinatorial auctions. In *Proc. 35th Intl. Colloq. on Automata, Languages and Programming*, pages 820–832, 2008.
- [24] E. H. Clarke. Multipart pricing of public goods. *Public Choice*, 11:17–33, 1971.
- [25] P. Cramton, Y. Shoham, and R. Steinberg (eds.). *Combinatorial Auctions*. MIT Press, Cambridge, MA, 2005.
- [26] C. Daskalakis, P. Goldberg, and C. Papadimitriou. The complexity of computing a nash equilibrium. In *Proc. 37th ACM Symp. on Theory of Computing*, 2006.

- [27] P. Dhangwatnotai, S. Dobzinski, S. Dughmi, and T. Roughgarden. Truthful approximation schemes for single-parameter agents. In *Proc. 49th IEEE Symp. on Foundations of Computer Science*, 2008.
- [28] S. Dobzinski. Two randomized mechanisms for combinatorial auctions. In *Proc. 11th Intl. Workshop on Approximation Algorithms for Combinatorial Optimization Problems*, 2008.
- [29] S. Dobzinski. An impossibility result for truthful combinatorial auctions with submodular valuations. In *Proc. 42nd ACM Symp. on Theory of Computing*, 2011.
- [30] S. Dobzinski and N. Nisan. Limitations of vcg-based mechanisms. In *Proc. 38th ACM Symp. on Theory of Computing*, 2007.
- [31] S. Dobzinski and N. Nisan. Mechanisms for multi-unit auctions. In *Proc. 9th ACM Conf. on Electronic Commerce*, 2007.
- [32] S. Dobzinski, N. Nisan, and M. Schapira. Approximation algorithms for combinatorial auctions with complement-free bidders. In *Proc. 36th ACM Symp. on Theory of Computing*, 2005.
- [33] S. Dobzinski, N. Nisan, and M. Schapira. Truthful randomized mechanisms for combinatorial auctions. In *Proc. 37th ACM Symp. on Theory of Computing*, 2006.
- [34] S. Dobzinski and M. Sundararajan. On characterizations of truthful mechanisms for combinatorial auctions and scheduling. In *Proc. 10th ACM Conf. on Electronic Commerce*, 2008.
- [35] D. Dubhashi and A. Panconesi. *Concentration of Measure for the Analysis of Randomized Algorithms*. Cambridge University Press, 2009.

- [36] S. Dughmi and T. Roughgarden. Black-box randomized reductions in algorithmic mechanism design. In *Proc. 51st IEEE Symp. on Foundations of Computer Science*, 2010.
- [37] S. Dughmi, T. Roughgarden, and Q. Yan. From convex optimization to randomized mechanisms: Toward optimal combinatorial auctions. In *Proc. 42nd ACM Symp. on Theory of Computing*, 2011.
- [38] B. Edelman, M. Ostrovsky, and M. Schwarz. Internet advertising and the generalized second price auction: Selling billions of dollars worth of keywords. In *Stanford Graduate School of Business Research Paper No. 1917*, 2005.
- [39] J. Feigenbaum, C. Papadimitriou, and S. Shenker. Sharing the cost of multicast transmissions. *Journal of Computer and System Sciences*, 63(1):21–41, 2001.
- [40] M. Gairing, B. Monien, and K. Tiemann. Selfish routing with incomplete information. In *Proc. 17th ACM Symp. on Parallel Algorithms and Architectures*, 2005.
- [41] M. Goemans, V. Mirrokni, and A. Vetta. Sink equilibria and convergence. In *Proc. 46th IEEE Symp. on Foundations of Computer Science*, 2005.
- [42] R. Gonen and D. Lehmann. Optimal solutions for multi-unit combinatorial auctions: Branch and bound heuristics. In *Proc. 2nd ACM Conf. on Electronic Commerce*, 2000.
- [43] R. Gonen and D. Lehmann. Linear programming helps solving large multi-unit combinatorial auctions. In *Electronic Market Design Workshop*, 2001.
- [44] T. Groves. Incentives in teams. *Econometrica*, 41:617–631, 1973.
- [45] J. Hannan. Approximation to bayes risk in repeated plays. In M. Dresher, A. Tucker, and P. Wolfe, editors, *Contributions to the Theory of Games*, volume 4. Princeton University Press, 1957.

- [46] J. C. Harsanyi. Games with incomplete information played by ‘bayesian’ players, parts i ii and iii. *Management Science*, 14, 1967-68.
- [47] J. Hartline, R. Kleinberg, and A. Malekian. Bayesian incentive compatibility via matchings. In *Proc. 22nd ACM Symp. on Discrete Algorithms*, 2011.
- [48] J. Hartline and B. Lucier. Bayesian algorithmic mechanism design. In *Proc. 41st ACM Symp. on Theory of Computing*, 2010.
- [49] J. Hastad. Clique is hard to approximate within $n^{1-\epsilon}$. *Electronic Colloquium on Computational Complexity*, 38, 1997.
- [50] J. Hastad. Some optimal inapproximability results. In *Proc. 29th ACM Symp. on Theory of Computing*, 1997.
- [51] R. Holzman, N. Kfir-Dahav, D. Monderer, and M. Tennenholtz. Bundling equilibrium in combinatorial auctions. *Games and Economic Behavior*, 47:104–123, 2004.
- [52] S. Horn. One-pass algorithms with revocable acceptances for job interval selection. University of Toronto MSC thesis, 2004.
- [53] N. Immorlica, D. Karger, E. Nikolova, and R. Sami. First-price path auctions. In *Proc. 7th ACM Conf. on Electronic Commerce*, 2005.
- [54] N. Immorlica and B. Lucier. On the impossibility of black-box truthfulness without priors. In *Workshop on Bayesian Mechanism Design*, 2011.
- [55] M. Jackson. A crash course in implementation theory. *Social Choice and Welfare*, 18:655–708, 2001.
- [56] S. Kakade, A. Kalai, and K. Ligett. Playing games with approximation algorithms. In *Proc. 38th ACM Symp. on Theory of Computing*, 2007.

- [57] A. Kalai and S. Vempala. Efficient algorithms for online decision problems. *Journal of Computer and System Sciences*, 71(3):291 – 307, 2005. Learning Theory 2003.
- [58] S. Khot, R. Lipton, E. Markakis, and A. Mehta. Inapproximability results for combinatorial auctions with submodular utility functions. In *Workshop on Internet and Network Economics (WINE)*, 2005.
- [59] P. Krysta. Greedy approximation via duality for packing, combinatorial auctions and routing. In *Proc. 30th Intl. Symp. on Mathematical Foundations of Computer Science*, 2005.
- [60] R. Lavi, A. Mu’alem, and N. Nisan. Towards a characterization of truthful combinatorial auctions. In *Proc. 44th IEEE Symp. on Foundations of Computer Science*, 2003.
- [61] R. Lavi and N. Nisan. Online ascending auctions for gradually expiring goods. In *SODA05*, 2005.
- [62] R. Lavi and C. Swamy. Truthful and near-optimal mechanism design via linear programming. In *Proc. 46th IEEE Symp. on Foundations of Computer Science*, 2005.
- [63] B. Lehmann, D. Lehmann, and N. Nisan. Combinatorial auctions with decreasing marginal utilities. In *Proc. 3rd ACM Conf. on Electronic Commerce*, 2001.
- [64] D. Lehmann, L. I. O’Callaghan, and Y. Shoham. Truth revelation in approximately efficient combinatorial auctions. In *Proc. 1st ACM Conf. on Electronic Commerce*, pages 96–102. ACM Press, 1999.
- [65] R. Paes Leme and E. Tardos. Sponsored search equilibria for conservative bidders. In *Fifth Workshop on Ad Auctions*, 2009.

- [66] B. Lucier. Beyond equilibria: Mechanisms for repeated combinatorial auctions. In *Proc. 1st Symp. on Innovations in Computer Science*, 2010.
- [67] B. Lucier and A. Borodin. Price of anarchy for greedy auctions. In *Proc. 21st ACM Symp. on Discrete Algorithms*, 2010.
- [68] M. Mahdian and A. Saberi. Multi-unit auctions with unknown supply. In *Proc. 8th ACM Conf. on Electronic Commerce*, 2006.
- [69] A. Mas-Colell, M. Whinston, and J. Green. *Microeconomic Theory*. Oxford University Press, 1995.
- [70] J. Mestre. Greedy in approximation algorithms. In *Proc. 14th European Symp. on Algorithms*, 2006.
- [71] A. Mu'alem and N. Nisan. Truthful approximation mechanisms for restricted combinatorial auctions. *Games and Economic Behavior*, 64:612–631, 2008.
- [72] R. Myerson. Optimal auction design. *Mathematics of Operations Research*, 6:58–73, 1981.
- [73] R. Myerson. Harsanyi's games with incomplete information. In *Management Science*, 2004.
- [74] N. Nisan. Bidding and allocation in combinatorial auctions. In *Proc. 2nd ACM Conf. on Electronic Commerce*, 2000.
- [75] N. Nisan. The communication complexity of approximate set packing and covering. In *Proc. 29th Intl. Colloq. on Automata, Languages and Programming*, 2002.
- [76] N. Nisan and A. Ronen. Algorithmic mechanism design. In *Proc. 31st ACM Symp. on Theory of Computing*, pages 129–140. ACM Press, 1999.

- [77] N. Nisan, T. Roughgarden, É. Tardos, and V. Vazirani, editors. *Algorithmic Game Theory*. Cambridge University Press, 2007.
- [78] R. Paes Leme and B. Lucier. Improved social welfare bounds for gsp at equilibrium. In *Proc. 13th ACM Conf. on Electronic Commerce*, 2011.
- [79] C. Papadimitriou. Algorithms, games and the internet. In *Proc. 33rd ACM Symp. on Theory of Computing*, pages 749–752. ACM Press, 2001.
- [80] C. Papadimitriou, M. Schapira, and Y. Singer. On the hardness of being truthful. In *Proc. 49th IEEE Symp. on Foundations of Computer Science*, 2008.
- [81] K. Roberts. The characterization of implementable choice rules. In J. Laffont, editor, *Aggregation and Revelation of Preferences*, chapter 18, pages 321–47. North Holland Publishing Company, 1979.
- [82] T. Roughgarden. Intrinsic robustness of the price of anarchy. In *Proc. 40th ACM Symp. on Theory of Computing*, 2009.
- [83] T. Roughgarden and E. Tardos. How bad is selfish routing? *Journal of the ACM*, 2002.
- [84] T. Sandholm. Algorithm for optimal winner determination in combinatorial auctions. *Artificial Intelligence*, 135:1–54, 2002.
- [85] Hal R. Varian. Position auctions. *International Journal of Industrial Organization*, 25(6):1163 – 1178, 2007.
- [86] W. Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *J. of Finance*, 16:8–37, 1961.
- [87] D. Zuckerman. Linear degree extractors and the inapproximability of max clique and chromatic number. In *Proc. 37th ACM Symp. on Theory of Computing*, 2006.